webdyn
Flexitron group

WebdynSunPM
Manual

User Manual

Table of contents

1 Introduction 9
1.1 Introduction 9
1.1.1 Description of WebdynSunPM 9
1.1.2 Key Capabilities of the Equipment 10
1.2 Overview of the Equipment 11
1.3 Software Overview 12
1.3.1 Functional Architecture 13
1.3.2 Data Acquisition and Extraction 14
1.3.3 Metadata Enrichment 14
1.3.4 Communication with Remote Servers 14
1.3.5 Script Engine 15
1.3.6 Administration and Configuration 15
1.4 Product Reference 15
2 Specifications 15
2.1 Technical Specifications 16
2.1.1 General Specifications 16
2.1.2 Technical Specifications 17
2.1.3 Cellular network band 17
2.1.4 Network Specifications 18
2.1.5 Regulatory Compliance 18
2.2 Hardware 18
2.2.1 Power Supply 19
2.2.2 Internal Battery 19
2.2.3 Cellular Connectivity 19
2.2.3.1 Modem 19
2.2.3.2 Antenna 20
2.2.3.3 SIM Card 20
2.2.4 Ethernet Interfaces LAN1/2 20
2.2.5 RS-485/422 Serial 1/2/3 Interfaces 22
2.2.6 Inputs 23
2.2.6.1 0-10 V/4-20 mA analog inputs 23
2.2.6.2 Digital inputs (dry contact or pulse) S0 24
2.2.7 Relay Output 25
2.2.8 SD Card 26
2.2.9 USB Interface 27
2.2.10 Connectors 27
2.2.11 Buttons and Status LEDs 28
2.2.11.1 POWER button 28
2.2.11.2 FACTORY RESET button 29
2.2.11.3 Status LEDs 29

2.3 Plant and Device Manager 31
2.3.1 Power Plant 31
2.3.1.1 Concept 31
2.3.1.2 Node 31
2.3.2 Equipment 32
2.3.2.1 Object model 32
2.3.2.2 Attributes 33
2.3.3 Definition Files 35
2.3.3.1 Role 35
2.3.3.2 File format 35
2.3.3.3 Sources of the definition files 35
2.4 Field Communications Manager 41
2.4.1 General Principle 41
2.4.2 Supported Protocols 42
2.4.3 Bus Management 43
2.4.4 Principle of Communication 43
2.4.4.1 Polling 43
2.4.4.2 Handling Request Failures 44
2.4.4.3 Multi-bus management 44
2.4.4.4 Integration of External Requests 44
2.4.4.5 Data Availability 44
2.4.5 Error Handling 45
2.4.5.1 Types of Errors 45
2.4.5.2 Behavior in the Event of a Request Error 45
2.4.5.3 Impact on Data 45
2.4.5.4 Equipment Communication Status 46
2.5 Remote Connection Manager 46
2.5.1 Role 46
2.5.2 Ethernet 46
2.5.3 Mobile 47
2.5.4 Behavior 47
2.5.5 Global Connectivity 47
2.5.6 Selecting the Communication Path 47
2.6 Remote Server Manager 48
2.6.1 Role 48
2.6.2 Server Configuration 48
2.6.3 Planning 49
2.6.4 What to Do in Case of Failure 49
2.6.5 Behavior Based on Network Connectivity 49
2.6.6 MQTT Behavior 49
2.6.7 Oversight and Bylaws 50

2.7 System Manager 50
2.7.1 Firmware 50
2.7.1.1 Software Version 50
2.7.1.2 Firmware Update 50
2.7.1.3 Catering 51
2.7.1.4 Limitations 52
2.7.2 Local System Settings 52
2.7.2.1 Date and Time 52
2.7.2.2 Time zone 52
2.7.2.3 Product ID 53
2.7.2.4 Persistent settings 53
2.7.2.5 Admission Rules 54
2.7.3 System Maintenance 54
2.7.3.1 Restart 54
2.7.3.2 Return to Factory 55
2.7.3.3 Automatic Recovery 55
2.7.3.4 Diagnostics and System Status 55
2.7.4 Logs 56
2.7.5 Licenses 56
2.7.5.1 License Status 56
2.7.5.2 Obtaining a License 57
2.7.5.3 Installation and Updates 57
2.7.5.4 Persistence 57
2.8 Modbus TCP Slave 57
2.8.1 General Operation 57
2.8.2 Read or write a variable 58
2.8.3 Execute a command 58
2.8.4 Predefined Webdyn Variables 58
2.8.5 User variables 59
2.8.6 Modbus Error Handling 59
2.9 Web Services 60
3 Installation 61
3.1 Safety Instructions 61
3.1.1 Electrical Connection 62
3.1.2 Environment 62
3.1.3 Chemical agent 62
3.2 Unboxing 63
3.2.1 Contents 63
3.2.2 Identification 63
3.3 Installation 65
3.3.1 DIN-rail mounting 65
3.3.2 Antenna 66
3.3.3 SIM Card 68

3.3.4 Environment 68
3.4 Wiring 68
3.4.1 Connectors 68
3.4.2 RS485/422 Serial Connection 69
3.4.2.1 "Daisy-Chain" Topology 69
3.4.2.2 Cable Type 70
3.4.2.3 120-ohm termination 70
3.4.2.4 2-wire RS485 wiring (half-duplex) 70
3.4.2.5 4-wire RS485 wiring (full-duplex) 71
3.4.2.6 Shielding 72
3.4.2.7 Distance 72
3.4.2.8 Maximum number of devices 72
3.4.3 0-10V or 4-20mA analog inputs 73
3.4.4 S0 pulse inputs or dry-contact inputs 73
3.4.5 Relay Output 74
4 Commissioning 75
4.1 First Startup 75
4.1.1 Logging In to the Web Interface 75
4.1.1.1 Changing the IP Address in Windows 10 77
4.1.1.2 Changing the IP Address in Windows 11 80
4.1.2 Quick Setup 82
4.2 Configuring SFTP/FTP/WebDAV Servers 82
4.2.1 Authentication and Security 82
4.2.2 Tree Structure 83
4.3 Web Server 84
4.3.1 Configuring Network Interfaces 84
4.3.1.1 Ethernet (local) 84
4.3.1.2 Mobile (modem) 85
4.3.2 Configuration of the Main and Secondary Servers 87
4.3.2.1 Interface and Protocol 88
4.3.2.2 SFTP/FTP/WebDAV Server 88
4.3.2.3 MQTT/MQTTs/MQTT on Azure/MQTT on AWS 92
4.3.2.4 SD card configuration 95
4.3.2.5 Planning Server Connections 97
4.3.3 Device Configuration 98
4.3.3.1 Automatic Detection 99
4.3.3.2 Manual Management of Devices 102
4.3.3.3 Edit/Duplicate/Delete Device 104
4.3.3.4 Edit Internal I/Os 105
4.3.4 Configuring Serial Connections 107
4.3.5 Equipment Definition File Library 108
4.3.5.1 Library Update 108
4.3.5.2 Library Management 109

4.3.6 Script Library 111
4.3.6.1 Library Update 112
4.3.6.2 Library Management 112
4.4 Commissioning by the main server 116
4.4.1 Configuration Files 116
4.4.1.1 Establishing the Connection 117
4.4.1.2 Behavior Upon First Connection 117
4.4.1.3 Behavior at Rated Conditions 118
4.4.2 Useful Commands 118
4.4.2.1 Start a SunSpec Automatic Discovery 118
4.4.2.2 Retrieve the gateway's mobile number 118
4.5 Activation via SMS 119
4.5.1 APN Settings 119
4.5.2 Configuring the Primary Server for SFTP 120
4.5.3 Initiating the first connection to the main server 120
4.6 Modbus TCP Slave 121
4.6.1 Configuration by Primary Server 121
4.6.2 Configuration via the Web Server 122
4.6.3 Definition File 122
4.7 Setup via SD Card 123
5 Operation 123
5.1 SFTP/Webdav Server 123
5.1.1 File system 123
5.1.1.1 Tree Structure 123
5.1.1.2 File Exchange and Synchronization 124
5.1.2 Gateway Configuration 125
5.1.2.1 Update the definition file library 125
5.1.2.2 Update the license file 126
5.1.3 Analyzing the collected data 126
5.1.3.1 File format 126
5.1.3.2 Contents 127
5.1.3.3 Data 127
5.1.4 Using Alarms 130
5.1.4.1 File Format and Synchronization 130
5.1.4.2 Alarms Generated by the Equipment 131
5.1.4.3 IO/Modbus Alarms 131
5.1.4.4 The Case of Virtual Equipment 132
5.1.5 Send Orders 132
5.1.6 Using Scripts 133
5.1.6.1 List of Available Scripts 133
5.1.6.2 Add scripts 133
5.1.6.3 Configure and Run a Script 133
5.1.6.4 Running a script 134

5.2 MQTT 134
5.2.1 Using the MQTT Service 134
5.2.1.1 Configuration 134
5.2.1.2 Server Directory Tree 135
5.2.2 Using Alarms 135
5.2.2.1 Format 135
5.2.2.2 Examples 137
5.2.3 Leveraging Data 137
5.2.3.1 Format 137
5.2.3.2 Examples 138
5.2.4 Send Orders 141
5.3 Script License Management 141
5.3.1 License Generation 142
5.3.2 License Activation via the Built-in Web Server 142
5.3.3 License Activation by the Primary Server 142
6 Maintenance 143
6.1 Firmware Update 143
6.1.1 Through the web interface 143
6.1.2 Via SD card 145
6.1.3 Via FTP/SFTP/WebDAV 145
6.1.4 Via SMS, MQTT, or FTP/SFTP commands 146
6.2 Changes to System Settings 146
6.3 Restart and Factory Reset 148
6.3.1 Restart and Shutdown 148
6.3.2 Return to Factory 149
6.4 Diagnostics 150
6.4.1 Equipment Communication 150
6.4.1.1 Equipment Status 150
6.4.1.2 Communication problem with the equipment 151
6.4.2 Remote Communication 152
6.4.2.1 Modem Connection Diagnosis 153
6.4.2.2 Ethernet Connection Diagnosis 154
6.5 Logs 155
6.5.1 Log Files 155
6.5.1.1 System Information and NTP Synchronization 156
6.5.1.2 Sending Alarms and Data 156
6.5.1.3 Synchronizing Configuration Files 156
6.5.1.4 Script Synchronization 157
6.5.1.5 End of connection 157
6.5.2 Equipment Communication Diagnostic Logs 157
6.5.2.1 Serial and Ethernet Interface Logs 157
6.5.2.2 Log of Entries and Exits: 158
6.5.3 SunSpec Detection Logs 159

6.5.4 Script Logs 160
6.5.5 System Logs 161
7 Annexes 162
7.1 Configuration Files 162
7.1.1 <uid>_config.ini 163
7.1.1.1 Update File 163
7.1.1.2 gateway Information 163
7.1.1.3 NTP Clock Synchronization 163
7.1.1.4 Server File System Directory Structure 163
7.1.1.5 Server Configuration 164
7.1.2 <uid>_daq.csv 167
7.1.2.1 Modem Configuration 167
7.1.2.2 Ethernet Configuration 168
7.1.2.3 Serial Configuration 169
7.1.2.4 Modbus TCP Slave Configuration 170
7.1.2.5 Equipment Configuration 170
7.1.3 <UID>_license.ini 171
7.1.4 <UID>_scl.ini 172
7.1.5 <uid>_var.ini 173
7.2 External Commands 174
7.2.1 Order Reference 174
7.2.1.1 SYSTEM 174
7.2.1.2 NETWORK 182
7.2.1.3 MONITORING 186
7.2.1.4 INSPECTION 188
7.2.2 Remote Controls 190
7.2.2.1 Principle 190
7.2.2.2 List of External Commands 192
7.2.2.3 Command File (FTP, SFTP, WebDAV) 192
7.2.2.4 MQTT Command Message 194
7.2.2.5 Text Message 195
7.2.2.6 Modbus TCP Slave 196

1 Introduction

1.1 Introduction

1.1.1 Description of WebdynSunPM

WebdynSunPM is an IoT edge gateway designed for the smart grid and photovoltaic sectors.

It serves as the central data and control gateway for the facility, collecting field data, organizing it, and enabling remote monitoring and energy management.

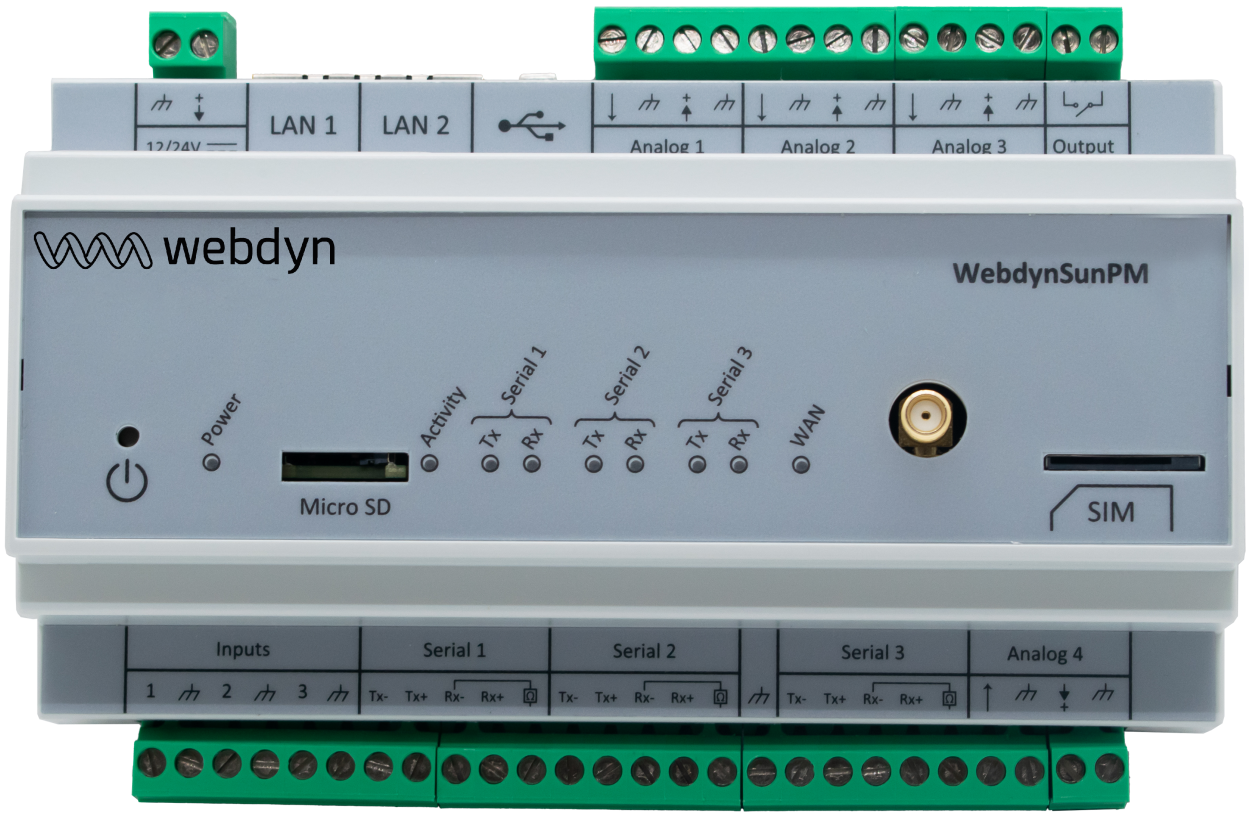
The gateway interfaces with a wide range of field equipment, such as inverters, batteries, energy meters, weather sensors, and electric vehicle charging stations.

By providing a central collection point, data is standardized and accessible in a single location, thereby replacing multiple proprietary devices.

The concentrator collects all data from field devices and aggregates it to enable remote operation of the facility. The concentrator is compatible with devices that communicate via Modbus TCP/RTU or proprietary manufacturer protocols.

WebdynSunPM enables operators to:

- **Analyze** the performance and productivity of their facility
- **Monitor** the operational status of the equipment
- **Manage** energy flows



Product Picture

1.1.2 Key Capabilities of the Equipment

Field data acquisition

The gateway collects data from devices using standard industrial protocols such as Modbus TCP/RTU, as well as proprietary manufacturer protocols.

The collected data is structured so that it can be used by remote monitoring or analysis systems.

Remote communication

Communication with remote systems is provided by:

- a built-in 4G modem,
- an Ethernet connection to a third-party router.

The gateway can connect to up to two remote servers (primary and secondary) to synchronize configuration files and transmit the collected data.

Data exchanges are based on open file formats (such as CSV) that ensure interoperability with third-party tools.

Cybersecurity

Communications with remote servers can use secure protocols such as:

- HTTPS / WebDAV
- SFTP
- MQTTS

Insecure protocols (FTP, MQTT) may also be used when interoperability with existing systems requires it.

Administration and configuration

The gateway can be configured:

- locally via an embedded web interface accessible over Ethernet,
- remotely via file transfers with the server.

A configurable scheduler allows you to organize connections to remote servers for file synchronization and data transfer.

Energy management and script engine

The product includes a scripting engine that allows application code to be executed to interact with connected devices, particularly in the context of energy management:

- power control,
- decoupling,
- battery management,
- other specific approaches.

This engine is open-source and well-documented, allowing users to develop their own scripts or use the library provided by Webdyn.

SMS management

When the product is equipped with a provisioned SIM card, it can be queried and configured via text message. SMS commands are encrypted to ensure an appropriate level of security.

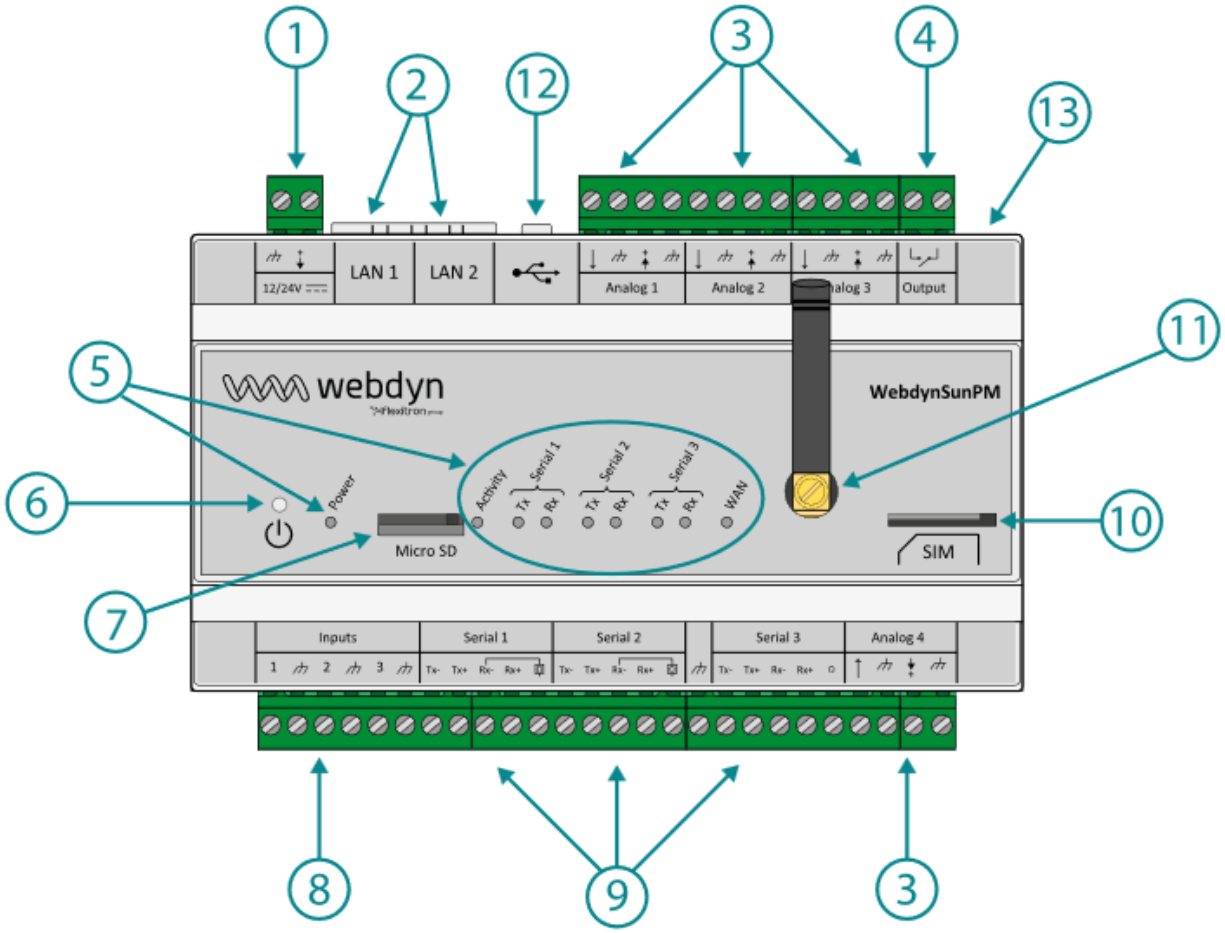
Software licenses

Some additional software features can be enabled through optional licenses.

1.2 Overview of the Equipment

The gateway is a network-enabled industrial device designed to be installed in the **Smart Grid** and **Renewable Energies** facilities as a photovoltaic power plant.

It includes all the physical interfaces needed to communicate with field equipment and remote systems.



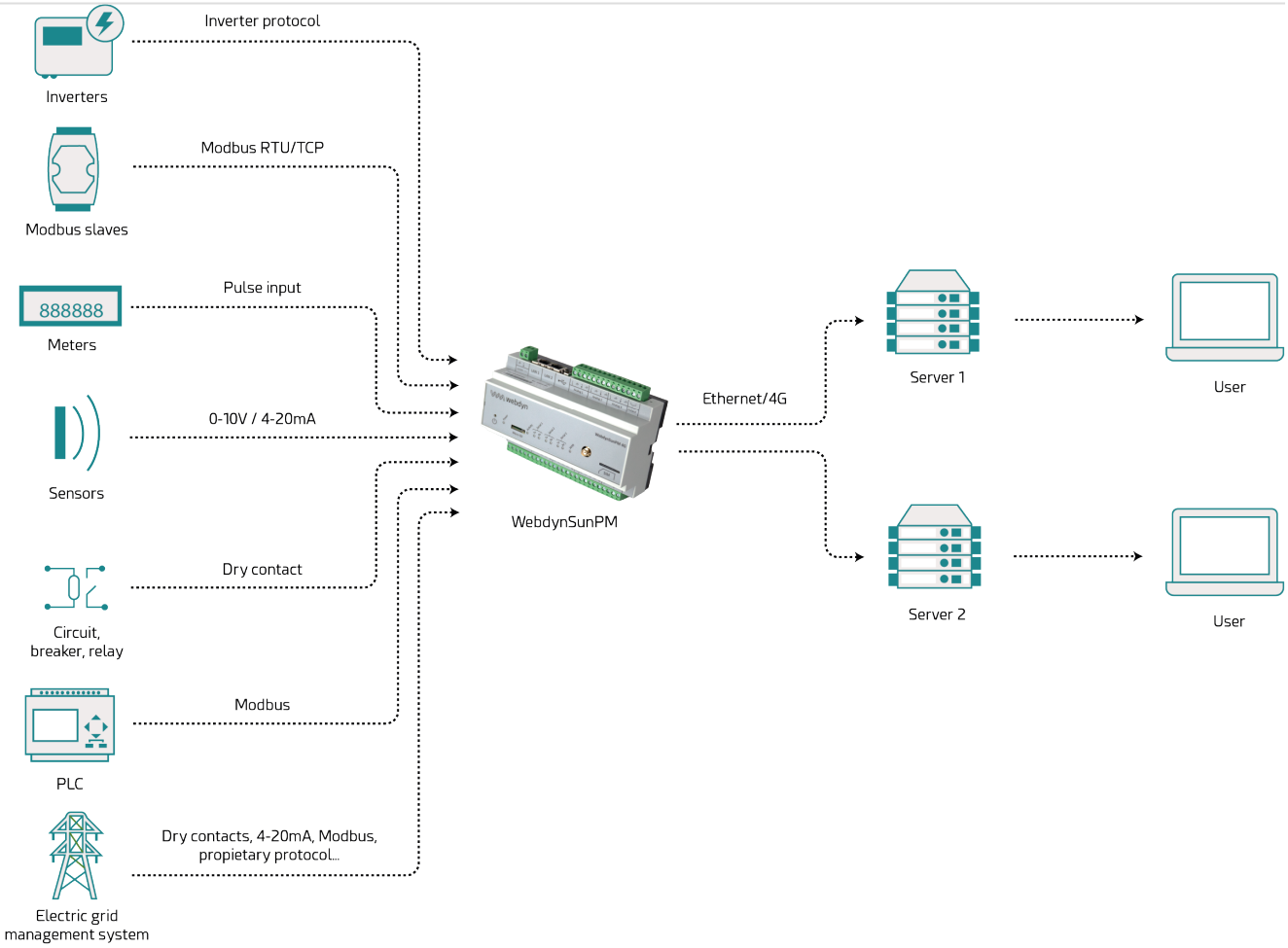
Hardware overview

Number	Description
1	12–24 V Power Terminal Block
2	2x Ethernet Ports
3	4x Analog Inputs 0–10 V or 4–20 mA
4	1x Relay Output (24V/1A)
5	Status Indicators
6	Power Button
7	Micro SD Card Slot
8	3x Digital Inputs (dry contact or S0 pulse)
9	3x RS485/RS422 Serial Ports
10	SIM Card Holder
11	SMA antenna connector on the modem
12	USB port
13	RESET Button

1.3 Software Overview

The software embedded in the gateway handles the acquisition of field data, its structuring, enrichment, and local processing before it is transmitted to remote systems.

It acts as an abstraction layer between physical equipment and monitoring or energy management applications.

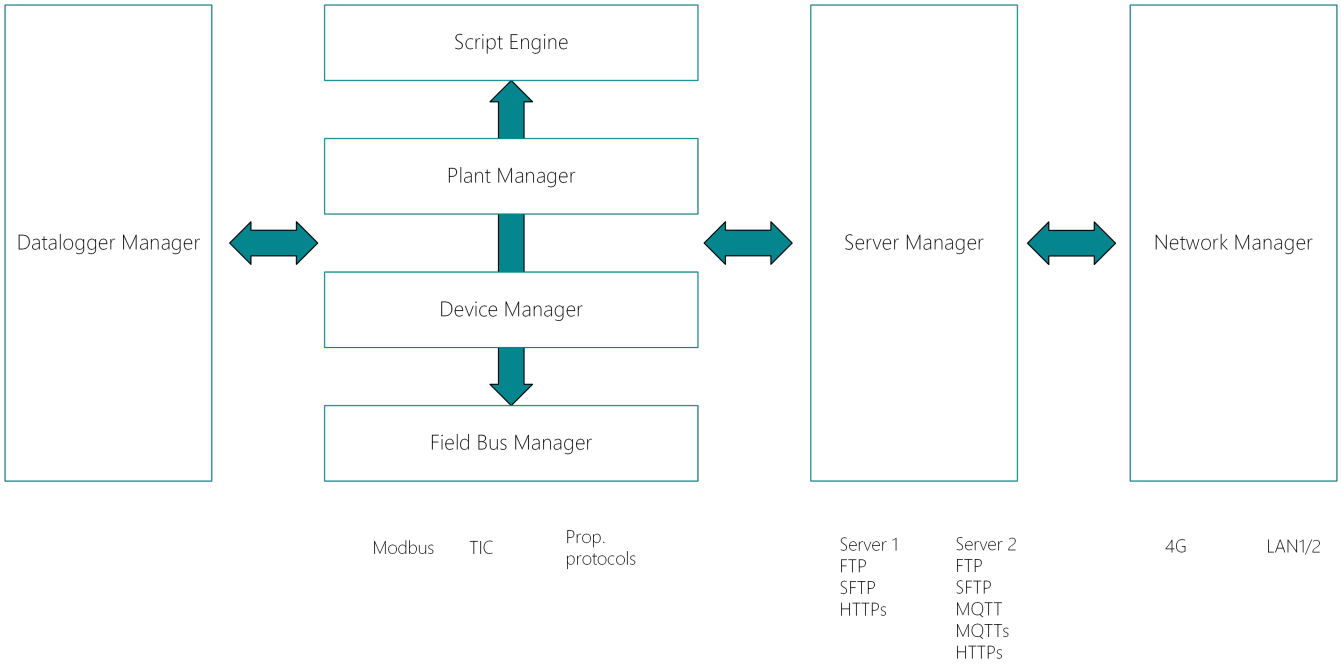


product architecture

1.3.1 Functional Architecture

The software is based on a modular architecture organized around the following building blocks:

- Field Data Collection
- Data Structuring and Enrichment
- Local Management and Application Logic
- Remote Communication
- Administration and Configuration



firmware architecture

1.3.2 Data Acquisition and Extraction

The software allows you to collect data from a variety of devices, such as:

- inverters
- batteries
- charging stations
- electric meters
- environmental sensors

Data is acquired via various industrial protocols or physical signals.

Regardless of their source, the data is processed according to a common model. Raw signals (registers, pulses, analog quantities) are abstracted so that they can be processed uniformly by the system.

1.3.3 Metadata Enrichment

To make the data immediately usable, the software associates metadata with the equipment and the collected variables.

In particular, this metadata makes it possible to:

- to identify the equipment
- to categorize them
- to associate a functional context with the data
- to standardize units and scales

This enrichment step is a central component of the software, as it transforms technical signals into information that can be used by remote applications.

1.3.4 Communication with Remote Servers

The gateway can connect to one or more remote servers in order to:

- transmit the collected data

- synchronize configuration files
- receive updates or orders

Transfers can be scheduled through an internal scheduling mechanism or initiated immediately based on operational needs.

Communications may use secure or unsecure protocols, depending on the integration environment.

1.3.5 Script Engine

The software includes a scripting engine **LUA** that allows you to execute specific application logic.

In particular, this engine allows for:

- to interact with connected devices
- to implement energy management strategies
- to implement custom rules

The engine is open-source and well-documented, allowing advanced users to develop their own scripts or use the provided libraries.

> **Info**
>
> A document specific to the LUA scripting engine is available at https://www.webdyn.com/

1.3.6 Administration and Configuration

The software can be configured:

- locally via the embedded web interface
- remotely via file transfers with Server 1

> **Note**
>
> Server 1 is designated as **serveur principal** and Server 2 as **serveur secondaire**. Only Server 1 allows remote modifications to the gateway.

This architecture enables centralized and scalable management of the facilities.

1.4 Product Reference

References	Descriptions
WG0517-A01	WebdynSunPM (Europe and India Version)
WG0517-A02	WebdynSunPM (Global Version)
WG0517-A03-DEIE	WebdynSunPM in a DEIE enclosure (European and Indian versions)
WG0517-A04	WebdynSunPM 4G (European and Indian Versions)

2 Specifications

2.1 Technical Specifications

2.1.1 General Specifications

Feature	Description
Power Supply	12...24 Vdc
Battery	650 mAh 3.7 V LiPo
Power	P: 5W (10W max)
Dimensions	155 x 106 x 58 mm, 9 DIN rail modules
Case	IP20, RoHS-compliant, DIN EN 60715 TH35, DIN VDE 0470-1, DIN 43880 Size 1, VBG 4, IEC 529
Weight	330 g
Operating Temperature	-5...+40°C
Storage Temperature	-20...+85°C
Relative humidity	25...75% RH (non-condensing)
Level of pollution	2
Regulations	CE - RED / RoHS / REACH

2.1.2 Technical Specifications

Feature	Description
Processor	ARM Cortex-A7 Core - 528 MHz
Memory	512 MB DDR3 SDRAM, 8 GB eMMC flash (50 MB per device)
Operating System	SunPM OS
2G cellular interface: EDGE, GSM, GPRS	850 MHz, 900 MHz, 1800 MHz, 1900 MHz
4G LTE cellular interface	B1, B3, B5, B7, B8, B20, and B28
SIM Format	Standard SIM, mini SIM, 2FF format, 1.8V/3V compatible
Ethernet Interface	2 10/100 Mbps ports
Serial interface	3x RS-485/422 2-4-wire
USB Interface	1 USB 2.0 port
Analog Inputs	4x 0–10 V or 4–20 mA (12-bit CAN)
Digital Inputs	3x dry or pulse contacts, S0, Class A/B
Relay	1x 24V/1A

2.1.3 Cellular network band

RF Band	Transmission Frequency	Max Power
GSM 850	824 MHz – 849 MHz	33 dBm (+-2 dB)
E-GSM 900	880 MHz – 915 MHz	33 dBm (+-2 dB)
DCS 1800	1710 MHz – 1785 MHz	30 dBm (+-2 dB)
PCS 1900	1850 MHz – 1910 MHz	30 dBm (+-2 dB)
LTE B1	1920 MHz–1980 MHz	23 dBm (+- 2 dB)
LTE Band 3	1710 MHz – 1785 MHz	23 dBm (+- 2 dB)
LTE Band 5	824 MHz – 849 MHz	23 dBm (+- 2 dB)
LTE B7	2500 MHz – 2570 MHz	23 dBm (+- 2 dB)
LTE B8	880 MHz – 915 MHz	23 dBm (+- 2 dB)
LTE B20	832 MHz – 862 MHz	23 dBm (+- 2 dB)
LTE Band 28	703 MHz – 748 MHz	23 dBm (+- 2 dB)

2.1.4 Network Specifications

Feature	Protocols & Format
Embedded Web Server	https
Secure Server Communication Protocol	sftp, mqtts, WebDAV-https
Server Communication Protocol	ftp, mqtt
Modbus Protocol	RTU, TCP
Clock Synchronization	remote NTP server
Data Exchange Format	CSV via FTP/SFTP/WebDAV-HTTPS, JSON via MQTT/MQTTs

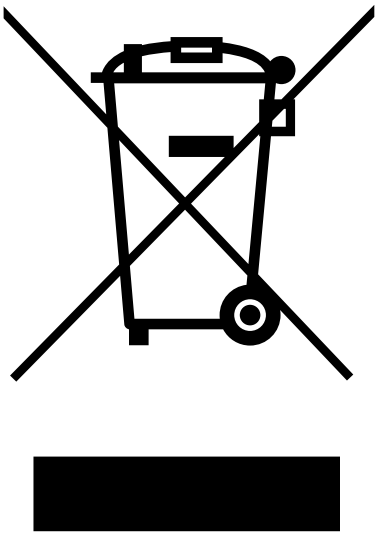
2.1.5 Regulatory Compliance

The product complies with European directives, as stated in the EU Declaration of Conformity available from Webdyn or at www.webdyn.com

Applicable guidelines:

- Directive 2014/53/EU (RED)
- 2011/65/EU (RoHS)

The product bears the following mandatory marking:

European Compliance	Waste Electrical and Electronic Equipment
CE	

2.2 Hardware

2.2.1 Power Supply

The concentrator is powered by an external 12..24-VDC power supply with a minimum power rating of 15W.

Warning

Review the installation and electromagnetic compatibility regulations in effect in your market.

Danger

The gateway may be damaged if the power supply polarity is not observed

2.2.2 Internal Battery

The gateway is equipped with a lithium (LiPo) battery to handle power outages:

- Send an alert in the event of a power outage
- Ensuring Data Integrity
- Secure Product Decommissioning

Note

The battery does not allow the product to be used during a power outage.

Warning

The battery charges automatically when the product is plugged in. In the event of frequent or prolonged power outages, the battery may not charge.

Danger

Using the product outside its operating temperature range may damage the battery, particularly at sub-zero temperatures

2.2.3 Cellular Connectivity

Modem

The gateway can be equipped with different cellular modems depending on the hardware version.

The differences mainly concern the supported network technology and the availability of carrier networks.

Modem Part Number	Supported Networks	Technology	Concentrator Reference
Sierra Wireless HL 8548	UMTS/HSPA+/GSM	3G (HSPA+)	WG0517-A01 / WG0517-A02
Quectel EG12Y	LTE + 2G fallback	4G (LTE CAT-1) / 2G	
Quectel EG915U	LTE Cat-1 bis	4G (LTE CAT-1 bis)	WG0517-A04

Antenna

The gateway is equipped with a female SMA connector on the front panel and comes with a 5-cm angled antenna. Below are the specifications for the included antenna/

Parameter	Specifications
Band	GSM850, GSM900, DCS, PCS, WCDMA I
Frequency	690–960/1710–2170/2400–2700 MHz
Impedance	50 ohms
Gain	2.5 dBi
Connector	Male SMA, Right Angle

SIM Card

The gateway has a SIM card slot that is directly accessible on the front panel.

To ensure proper operation, the SIM card and cell phone plan must have the following features:

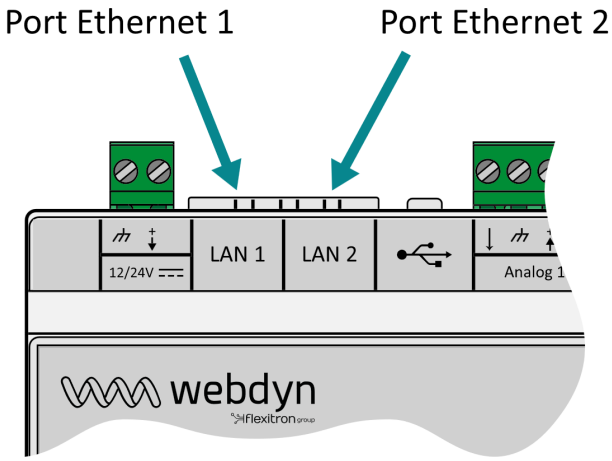
- Mini SIM 2FF format, 25x15mm
- Sending/Receiving Text Messages
- 2G/4G
- Minimum mobile data: 50 MB/month (depends on the number of devices and the number of connections to the servers)

Warning

Certain operations on the gateway may use a significant amount of mobile data, such as a firmware update (~5 to 10 MB). Make sure you have enough data allowance on your mobile plan.

2.2.4 Ethernet Interfaces LAN1/2

The gateway has 2 Ethernet ports (LAN1 and LAN2). These ports correspond to separate network interfaces.



ethernet interface

These interfaces allow the gateway to:

- communicate with local devices using Modbus TCP
- connect to a local network
- Establish a remote connection (Webdyn services, NTP synchronization, server 1/2 connection) via a router

These two interfaces are functionally similar and can be configured independently. They are preconfigured as shown in the following table:

Default setting	LAN1	LAN2
IP address	192.168.1.12	192.168.2.12
Subnet mask	255.255.255.0	255.255.255.0
Primary DNS	-	-
Secondary DNS	-	-
Footbridge	-	-

The web server is accessible via both interfaces.

Note

The web server is available via HTTP only on the local network. HTTP access is disabled if a gateway is configured or when accessing the server via the mobile network.

Note

It is not possible to assign an IP address via DHCP

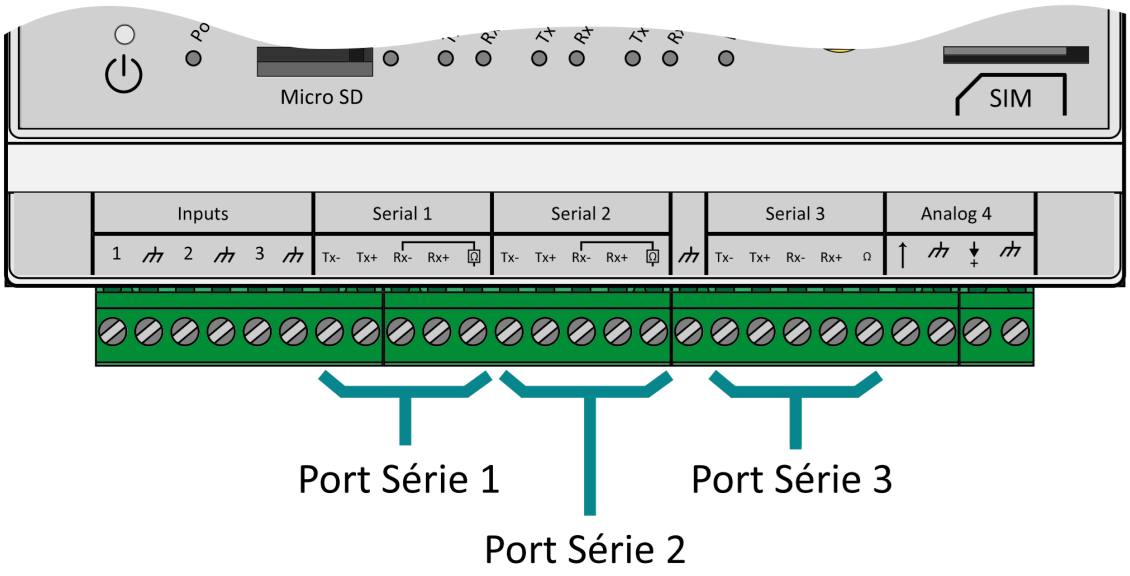
Each Ethernet port supports:

- A 10Base/100Base-Tx IEEE 802.3 connection

- Automatic signal detection and cross-referencing
- Automatic negotiation of speed (10/100 Mbps) and mode (half/full duplex).

2.2.5 RS-485/422 Serial 1/2/3 Interfaces

The gateway has 3 independent and configurable RS-485/422 serial interfaces.



serial interface

These interfaces enable communication with local devices that use Modbus RTU or a proprietary protocol.

The default configuration is listed in the table below:

Parameter	Episode 1/2/3
Fashion	**RS485 2 wires (defaut)**, 4-wire RS485
Baud rate	1200, 2400, 4800, **9800 (defaut)**, 19200, 38400, 57600, 115200, 230400, 460800
Data bits	7 ,**8 (defaut)**
Parity	**None (defaut)** , even , odd
Stop bit	**1 (defaut)**, 2

Additional settings are also available:

Parameter	Episode 1/2/3
Interframe (ms)	-
Forwarded TCP port	-

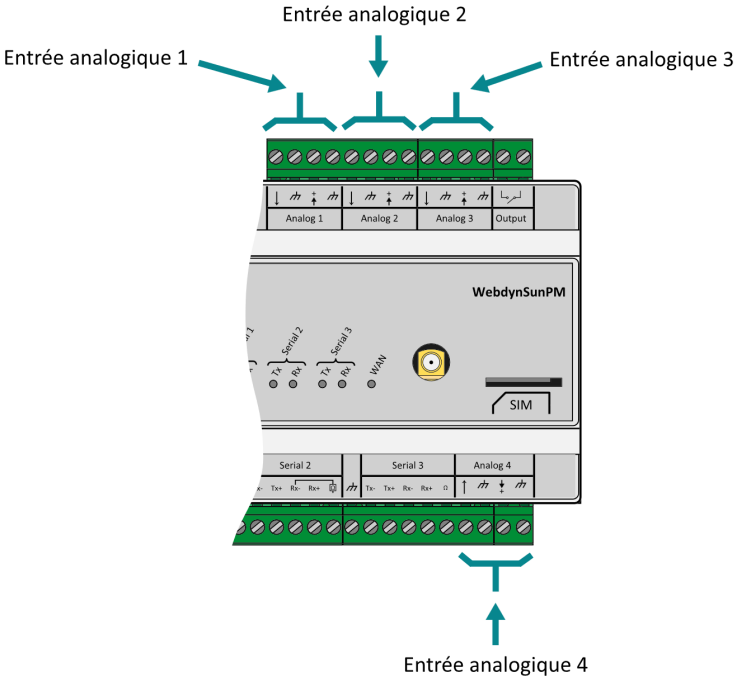
To ensure that a serial connection functions properly, a 120-ohm resistor must be placed at both ends of the line. The gateway can be placed anywhere along the serial connection and does not require an additional resistor, as this is provided by the cabling.

> **Warning**
>
> It is essential to follow best practices for serial connection cabling (cable type, length, speed, shielding) to avoid any risk of a failed or faulty connection. These rules are outlined in the Cabling section

2.2.6 Inputs

0-10 V/4-20 mA analog inputs

The concentrator has 4 analog inputs that can be configured for 0–10 V voltage or 4–20 mA current loops, allowing for the connection of analog sensors.



Analog inputs

The hub's analog-to-digital converters (ADCs) have a resolution of 12 bits:

- **Mode 0-10V**: 2.578 mV
- **Mode 4-20mA**: 5.249 µA

Each analog input can be wired as a 4-, 3-, or 2-wire connection using the 4-wire terminal block:

- **Supply 2 wires**
- **Signal 2 wires**

The supply voltage for the sensors is the same as the hub's supply voltage.

The grounds for all analog inputs are shared.

Warning

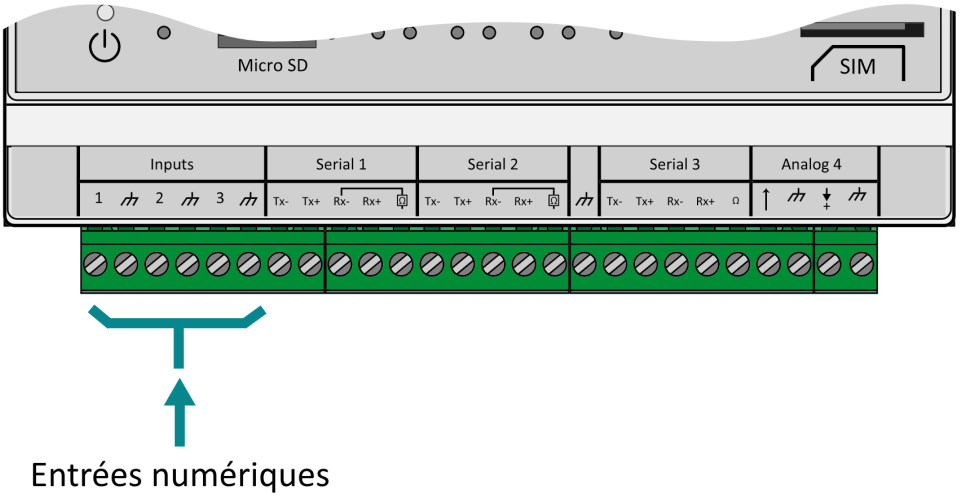
The total power of the sensors must not exceed 7W

Warning

Do not apply a voltage greater than 12V or a current greater than 24mA to the analog inputs

Digital inputs (dry contact or pulse) S0

The hub has 3 digital inputs that can be configured as dry contacts or S0 pulse counters.



Digital Inputs

Warning

The length must not exceed 100 meters

Dry contact

When the input is configured in dry contact mode, you can:

- Retrieve the status of a piece of equipment
- Trigger an alarm
- Interacting with the scripting engine

The characteristics of the digital input configured as a dry contact are listed below:

Feature	Value
Input Type	Dry contact - Active Low
Max. Voltage/Current	4 mA @ 5 V
Active Trigger Threshold "1"	< 1V
Inactive trigger threshold "0"	> 3.5V

Pulsed S0

The hub allows you to connect meters with Class A (24 V) and Class B (5 V) pulse outputs in accordance with the `CEI 62053-31:1998` standard. Class A is recommended for long-distance transmission.

The entry into S0 mode is of type **Sink**, meaning that a voltage is applied to the meter's S0+ terminal via an (internal) pull-up resistor, and a voltage of 0V is applied to the meter's S0- connection.

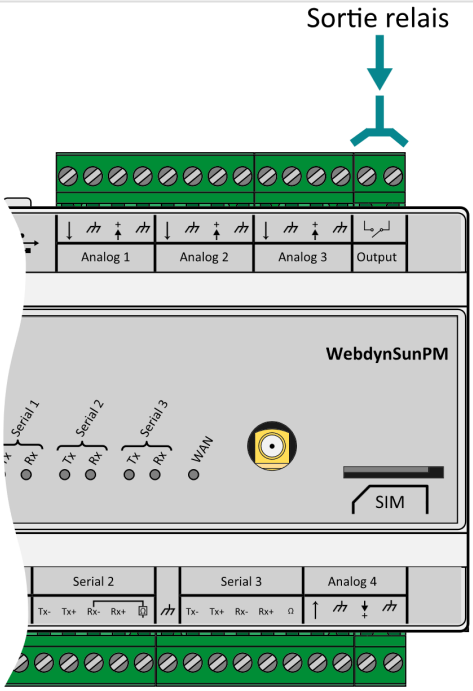
The specifications for the S0 pulse input are listed below:

Feature	Class A Value	Class B Value
Input Type	S0 Sink	S0 Sink
Internal Power Supply	24V	5V
Active trigger threshold	< 8 mA	< 1 mA
Trigger Threshold	> 15 mA	> 2.5 mA
Pulse width	> 20 ms	> 20 ms

> **Warning**
>
> For optimal Class A operation, the hub must be powered by 24 Vdc

2.2.7 Relay Output

The gateway has a potential-free relay output (dry contact)



relay output

The relay's specifications are:

Feature	Max Value
Voltage	24V
current	1A

Warning

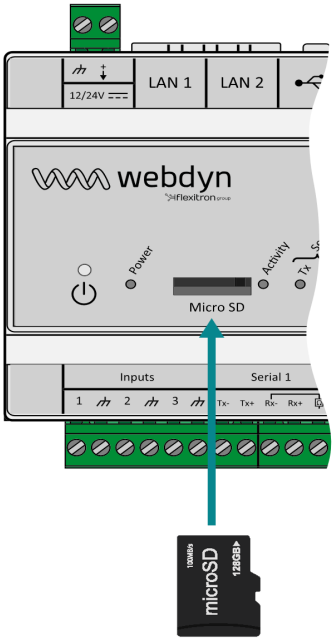
The hub's relay cannot be used to directly control high power. In this case, an external intermediate relay must be used.

The relay can be controlled by:

- File Transfer (SFTP, FTP, or WebDAV)
- MQTTS (MQTT) Message
- Text Message
- Scripts

2.2.8 SD Card

The gateway has an SD card slot on the front panel. Micro SDXC cards (15x11 mm) with a capacity of up to 32 GB are supported.



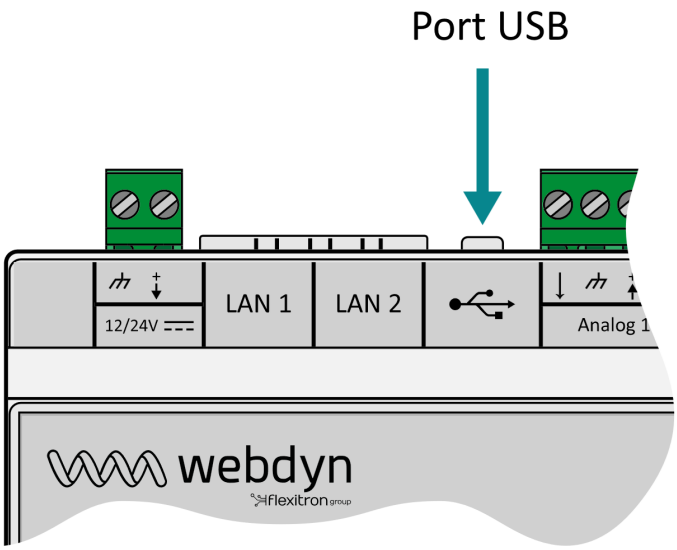
SD Card

The SD card can be configured as a remote server (Server 1) and can then be used to:

- Save the configuration
- Perform the update
- Store the data collected from the equipment

2.2.9 USB Interface

A USB port is available for connecting Webdyn TIC accessories for the remote reading of Enedis Linky and SME/SMI electricity meters.



usb port

2.2.10 Connectors

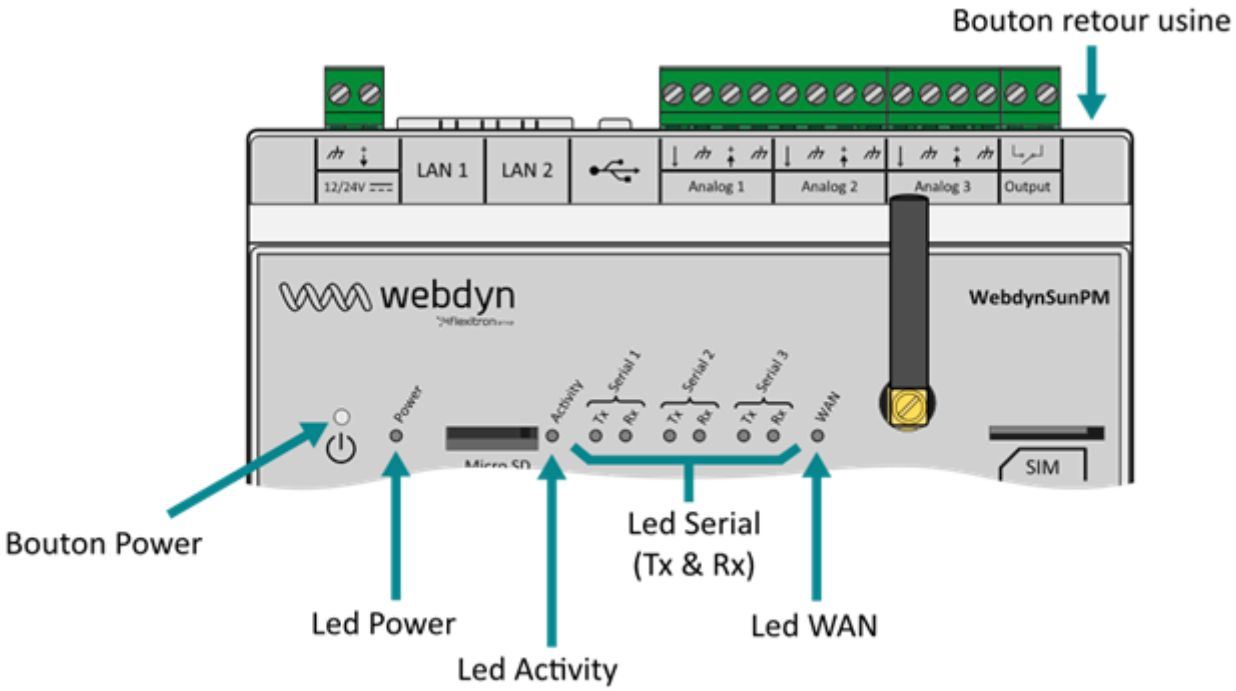
The gateway is equipped with male PCB connectors and comes pre-assembled with the corresponding female cable connectors.

The characteristics of female cable connectors are as follows:

Feature	Specifications
Connector Type	Screw-Cage Terminal Block
Cross-sectional area for a single-strand cable	0.05...2.5 mm²
Cross-sectional area for multi-strand cable	0.05...1.5 mm²
Maximum/Recommended Torque	0.6/0.5 Nm

2.2.11 Buttons and Status LEDs

The gateway is equipped with function buttons and status LEDs.



Buttons & Status LEDs

POWER button

The gateway starts up automatically when power is applied to the power terminal block.

You can turn it off or back on by pressing the **POWER** button, following these instructions:

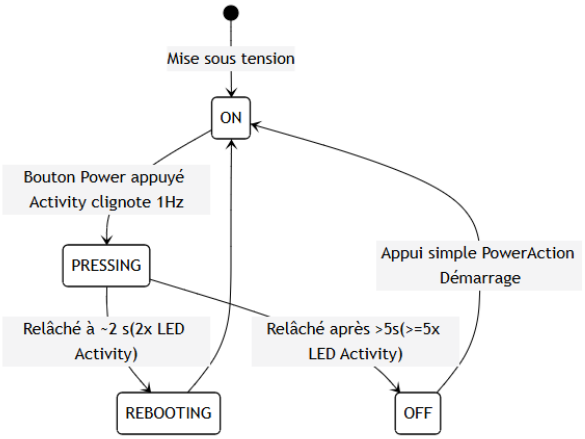


diagram state btn on

FACTORY RESET button

The **FACTORY RESET** button allows you to reconfigure the gateway if necessary (loss of IP address, loss of login credentials, etc.)

There are two types of **FACTORY RESET**:

- **IP Parameters Reset Only**
- **Gateway configuration Reset**

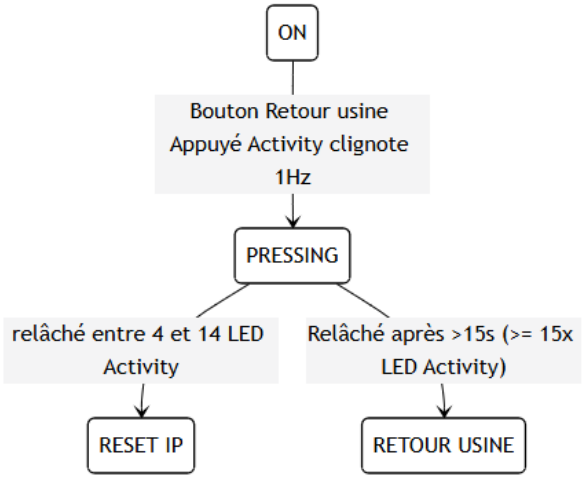


diagram state btn reset

Status LEDs

Status LEDs are located on the front panel of the gateway and also on the LAN1/2 connectors.

LED	Status	Meaning
POWER	Permanent	gateway in operation
	Off	gateway is turned off or has no power
ACTIVITY	Slow flashing	Normal operation
	Rapid flashing	Three possible cases
		A button pressed
		gateway update in progress
		Starting the gateway
SERIAL 1/2/3 Tx/ Rx	Flashing	Transmission (Tx) gateway -> device, Reception (Rx) device -> gateway
LAN1/2 LINK	Permanent	A physical connection to a piece of equipment exists
	Off	Cable Not Plugged In
LAN1/2 ACTIVITY	(Orange) Blinking	A communication is in progress
	Off	No communication in progress
LED WAN	see below	Server Connection Status

Note

In a 2-wire serial connection, the **SERIAL Rx** LED does not flash because data is received over the transmission line.

The **WAN** status LED is designed to help the user determine the connection status. This LED is configured based on Server 1 settings and displays three colors (green, orange, red).

Color	Status	Meaning
/	Off	No login attempts
Green	Flashing slowly	Last successful connection to the server
Green	Flashes rapidly	Connecting to the server...
Orange	Flashing slowly	NTP Synchronization Issue
Red	Flashing slowly	Problem connecting to the server

Note

If Server 1 is not configured, the **WAN** LED indicates the status of Server 2's MQTT connection.

2.3 Plant and Device Manager

2.3.1 Power Plant

Concept

> **Note**
>
> Since FW V5.2

A power plant is a unique top-level object. Its primary purpose is to provide a single point of interface for external stakeholders—such as aggregators, distribution system operators, and flexibility providers—to quickly understand the facility's performance and controllability.

The gateway defines a logical application object named **Plant** that organizes devices based on their functional relationships.

Each piece of equipment is associated with a node that represents a functional group.

A piece of equipment is therefore defined as:

- Its category
- Its node

The **Plant** object thus provides application methods for scripts and exposes aggregated data for each node.

Node

A node represents a functional energy domain. It is used to group together multiple pieces of equipment from different categories that have a meaningful use case in and of themselves. A node aggregates measurements, calculates KPIs, and provides unique control functions, all of which are related to its type.

A node belongs to a facility.

A node type is unique for each installation.

The node is therefore an attribute of the equipment.

The list of nodes is as follows:

- **SOLAR**: Photovoltaic Power Generation
- **GRID**: Connection point to the power distribution/transmission grid
- **CONSUMPTION**: Local consumer delivery point
- **STORAGE**: Storage
- **NONE**: Other

For example, the following equipment can be found:

- Inverter → SOLAR
- PCS Battery → STORAGE
- EMS Battery → STORAGE
- Meter at the connection point → GRID
- Electric Vehicle Charger → EV

- Consumer Meter → CONSUMPTION

Each node has methods specific to its application group. For example, here are the methods for the **SOLAR** node:

- **solarSetPowerOn**: Starting/Stopping the inverters
- **solarHasActiveControl**: Control of active power control capability
- **solarSetRatedPowerKW**: Setting the rated power
- **solarGetRatedPowerKW**: Get the rated power
- **solarSetActivePowerPercent**: Active power setting (%)
- **solarSetActivePowerKW**: Setting the active power in kW
- **solarGetActivePowerPercent**: Get the current active power as a percentage
- **solarGetActivePowerKW**: Get the current active power in kW
- **solarGetInvertersActivePowerKW**: Retrieve the current active power of the inverters

2.3.2 Equipment

An `Device` object represents a device that is monitored or controlled by the gateway.

It is the basic unit for:

- data acquisition
- order placement
- Integration into an application system (Plant / Node)

A `Device` is protocol-independent but depends on:

- from a definition file (`defFile`)
- a communication interface
- a protocol settings profile

Object model

```
Device:
  metadata:
    index: int
    name: string
    manufacturer: string |null
    model: string | null
    serialNumber: string | null
    category: enum
    node: enum
    plantIndex: int

  communication:
    interfaceId: enum
    address: int
    ipAddress: string | null
    ipPort: int | null
    timeout_ms: int
    parameters: object

  acquisition:
    acq_period_s: int
    def_file: string
    isReadOnly: boolean
    isProprietary: boolean

  output:
```

```
    data
    alarms
```

Attributes

The devices monitored by the gateway are varied:

- Modbus RTU/TCP
- Proprietary Inverter Protocols
- Modbus TCP slave
- Input/Output (I/O)
- ICT Customer Information System
- Virtual Equipment

> **Warning**
>
> Certain types, notably TIC and IO, use specific configuration profiles

Metadata

Attributes	Type	Description
index	int	Unique device identifier
name	string	Last Name
serialNumber	string	Serial Number (Read-Only)
category	enum	Device Type (from the definition file)
node	enum	Functional Affiliation
manufacturer	string | null	Manufacturer (from the definition file)
model	string | null	Template (from the definition file)
plantIndex	string | null	Application Membership Index (Plant Manager)

Category

- inverter
- meter
- sensor
- io
- logger
- extended values from definition files

> **Note**
>
> The list of categories is expanded whenever a definition file uses an unknown category

Node

- undefined

- none
- solar
- grid
- storage
- consumption

> **Note**
>
> If `node` is undefined, the system can automatically assign a default node based on the category (e.g., meter -> grid, inverter -> solar)

Communication

Attributes	Type	Description
interfaceId	enum	Communication Interface
address	int	Protocol address
ipAddress	string | null	IP address (Ethernet)
ipPort	int | null	TCP Port (Ethernet)
timeout_ms	int	Communication timeout
parameters	boolean	Protocol-specific

Acquisition

Attributes	Type	Description
acq_period_s	int	Acquisition Time
def_file	string	Definition file name
isReadOnly	boolean	Indicates whether the device can be modified
isProprierary	boolean	Indicates whether the protocol is proprietary

`defFile` defines:

- read/write registers
- data types
- conversion rules
- the associated alarms

> **Note**
>
> `Device` is a configured instance of `defFile`.

A `Device` produces two types of output:

- **measurement data**
- **alarms**

This data is then serialized according to the configuration of the publication channel (for example, CSV or JSON) and transmitted to remote servers.

> **Note**
>
> If the remote server is unreachable, the acquisition data is stored for 31 days or up to 50 MB of uncompressed data per device

2.3.3 Definition Files

Role

The definition file describes the data structure of a piece of equipment.

It allows you to:

- to standardize the data collected
- to define the accessible variables (measurements, states, commands)
- to apply conversion rules
- to classify the behavior of variables (measurement, alarm, parameter, etc.)

A definition file is associated with each piece of equipment (`Device`).

The following diagram shows the various sources of the definition files:

File format

The file is in `CSV` format with the separator `;` .

It consists of:

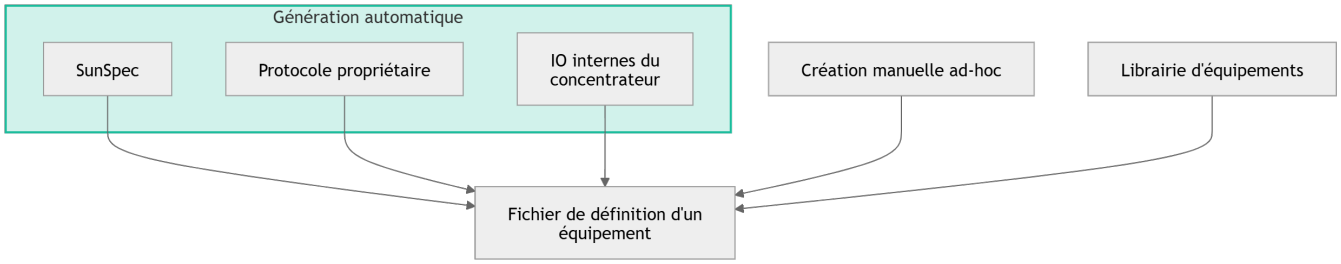
1. a header line describing the equipment
2. a list of variables

Sources of the definition files

Definition files can be obtained in several ways:

1. Manual Creation
2. From the bookstore **Webdyn**
3. Automatic Generation (SunSpec, Internal IO, Proprietary Protocols)

Below is a diagram illustrating the process:



def file source

Header

The first line of the file contains the following information:

`Protocol ; Category ; Manufacturer ; Model ; Forced writing code`

Protocol: Name of the protocol used. Possible values are:

- **modbus**: Modbus protocol over serial (RTU) or Ethernet (TCP) (FW > 5.1.0)
- **modbusRTU**: backward compatibility
- **modbusTCP**: backward compatibility
- **io**: Internal IO of the hub
- **tic**: Tic Protocol
- **ModbusSlave**: The hub is a Modbus slave device

Category: Equipment category—from a predefined, expandable list:

- **inverter**
- **meter**
- **sensor**
- **logger**
- **io**

> **Note**
>
> The category is used by application functions.

Manufacturer: Manufacturer Name

Model: Equipment model or variant

Forced writing code: Indicates whether the Modbus function code used for writing should be forced to 0x10. Possible values are:

- **0** (Default value):
  - Writing to a simple register is done using function code 0x06
  - Writing to multiple registers is done using function code 0x10
- **1**: Writing is performed only with function code 0x10

> **Note**
>
> The header of the file defining the hub's internal I/Os will then be:
>
> `io;Io;Webdyn;ioSunPM`

Following this first line, all subsequent lines will contain the definitions of the various equipment variables.

Each line fully defines a variable.

Each line will have the following format:

```
Index ; Info1 ; Info2 ; Info3 ; Info4 ; Name ; Tag ; CoefA ; CoefB ; Unit ; Action
```

- **Index** Unique identifier for the variable in the file. Used to identify variables in the data file, logs, or command files.
- **Info1**: Information specific to the protocol implemented on the device. See below
- **Info2**: Information specific to the protocol implemented on the device. See below
- **Info3**: Information specific to the protocol implemented on the device. See below
- **Info4**: Information specific to the protocol implemented on the device. See below
- **Name**: Unique variable name
- **Tag**: Standardized name that allows the use of scripts.
- **CoefA**: Coefficient to be applied to ensure the unit specified in the **unit** field is maintained. A floating-point number with the decimal separator `.`
- **CoefB**: Offset to apply. Floating-point number with the decimal separator `.`
- **Unit** Contains the desired unit. This field is for informational purposes only and may be empty
- **Action** Contains the processing code applied to this variable. The possible actions are listed below

Code Actions

Code	Description
0	Variable is disabled; it will not appear in the data files
1	Parameter; will not appear in the data files
2	Min / Max / Average
4	Instantaneous value
6	Instantaneous value; can be retrieved using the **GetData** command
7	Min / Max / Average; can be retrieved using the **GetData** command
8	Alarm (immediate activation)
9	Alarm (postponed until the next login)
10	Constant; does not appear in the data files

Note

Tags are used to collect variables of the same type and function from all devices. **Webdyn** tags are assigned to allow the hub to collect information from the devices and control them via scripts. The **Webdyn** tags are available in the appendices.

Tip

If a definition file exists and no tags have been applied, it is recommended to apply the **Webdyn** tags

> **Note**
>
> **CoefA** and **coefB** are used to transform the value so that it corresponds to the physical quantity being collected. These coefficients are applied to the values used by the scripts. If these values are missing, the default values are used:
>
> - **CoefA = 1**
> - **CoefB = 0**

Internal IO

Info1: I/O Type. Possible value described below.

Info1	Description
1	analog input (0–10 V or 4–20 mA)
2	digital input
3	switching relay output

Info2: I/O Number. Possible value described below.

If **Info1** = 1: Analog Input, possible values **Info2** = **1 à 4** If **Info1** = 1: Digital input; possible values: **Info2** = **1 à 3** If **Info1** = 1: relay output; possible values: **Info2** = **1**

Info3: I/O additional info

If **Info1** = 1, the possible values for **Info3** are:

- **1**: 4–20 mA analog input
- **2**: 0-10V analog input

If **Info1** = 2, the possible values for **Info3** are:

- **1**: digital input
- **2**: S0 pulse input

If **Info1** = 3, this field is not used

Info4: Not used

SunSpec: Automatic naming

The variables for a SunSpec device are generated based on the detected SunSpec tables:

- `<idTable>_<variableName>`: For each variable in the table declared in the SunSpec standards, a corresponding variable will be associated with the device. The variable name will consist of the table identifier followed by the variable name.
- `<idTable>_<repeatBlock>_<variableName>`: For variables that come from a repeating block, the variables are created using the table identifier, the repetition number, and the variable name, so that the generated name is unique. By default, the generated variables are of type `Paramètre` and will therefore have the code `Action` 1, with the exception of the variables in tables 101, 102, 103, 111, 112, 113, 123, 160, and 401, which will be created with the `Immédiat` type, i.e., code 4.

Info1: Register type

Info1	Description	Reading	Writing
1	coil	0x01	0x0F
2	discrete inputs	0x02	-
3	holding register	0x03	0x06 or 0x10
4	input	0x04	-

For holding-type registries `register`:

- Writing a value to a single register uses `0x06` by default;
- Writing a value to multiple registers uses `0x10`;
- If the "Forced writing code" header field is set to 1, the write operation uses `0x10` even for a value in a single register.

Info2: Address and size of register

- **Address**: Registry address.
  - Example: `40000` causes the registry to be read at the address `40000`
- **Address_size**: This format is used to specify the size of the data to be read, expressed in bytes. This format allows, for example, the reading of character strings. If the type is **U8** or **I8**, the **Taille** value corresponds to the offset to be applied to the register to retrieve the information.
  - Example 1: The value `40000_10` sets a variable whose data is located in register 40000 and is 10 bytes in size.
  - Example 2: For a value of type **I8**, the value `40000_1` indicates that you want to read the second byte from Modbus register 40000.
- **Adress_start bit_Nbits**. This format is used to specify the register at which a bit field begins, as well as the size of that bit field.
  - Example 1: `40005_4_8` configures a variable whose data is located in register `40005`, starting at bit 4, over an 8-bit length.
  - Example 2: For **Info1 = 3** or **Info1 = 4**, `40008_0_1` sets a variable of type **Bits** to read 1 bit from register `40008`, starting with bit 0.
  - Example 3: For **Info1 = 1** or **Info1 = 2**, the format remains `Adresse_0_Nbits`. The starting bit must be `0`. For example, `0_0_1` configures the read operation for a bit at address `0`. For **Info1 = 1** (coil), this variable can also be written; for **Info1 = 2** (discrete input), it is read-only.

Info3: Not supported

- **Unsigned integer**: `U8`, `U16`, `U32`, `U64`
- **Signed integer**: `I8`, `I16`, `I32`, `I64`
- **Float**: `F32`, `F64
- **String**: `String`
- **Bit fidl**: `Bits`
- **IP Address**: `IP` (4 bytes/2 registers)
- **IPv6 Address**: `IPV6` (16 bytes/8 registers)
- **MAC Address (format EUI48)**: `MAC` (6 bytes/3 registers)
- **Modbus Register** `RAW` (**Info2** of type **Adress_size**)

Info4 (SunSpec): Scale Factor

This field contains the name of the variable that determines its scale factor. When calculating the value of the configured variable, the decimal point of the read variable will be shifted by as many digits as the value of its **scale factor**. The formula is **var * 10sf**.

- Example 1: variable `var1=1234` with scale factor `sf_var1=3`; the decimal point of `var1` will be shifted 3 digits to the right to yield `1234000`
- Example 2: variable `var1=1234` with scale factor `sf_var1=-2`; the decimal point of `var1` will be shifted 3 digits to the right to yield `12,34`

Info4 (Modbus): Constante

When entered with the associated **code action 10**, the value is of type **Constant**. The value is then treated as a parameter and is not read from the bus.

> **Note**
>
> You can then add a new variable that will not be read via Modbus by leaving **info1** and **info2** blank.

Endianness

To manage endianness, **swap** modifiers can be added to the relevant values. List of allowed modifiers based on the value type:

- **_B**: Bytes are exchanged
- **_W**: Words are exchanged
- **_WB**: Words and bytes are exchanged

Type	Modifier	Conversion
U16, I16	**_B**	0xAABB→0xBBAA
U32, I32, F32	**_B**	0xAABBCCDD→0xBBAADDCC
U64, I64, F64	**_B**	0xAABBCCDDEEFF1122→0xBBAADDCCFFEE2211
U32, I32, F32	**_W**	0xAABBCCDD→0xCCDDAABB
U64, I64, F64	**_W**	0xAABBCCDDEEFF1122→0x1122EEFFCCDDAABB
U32, I32, F32	**_WB**	0xAABBCCDD→0xDDCCBBAA
U64, I64, F64	**_WB**	0xAABBCCDDEEFF1122→0x2211FFEEDDCCBBAA

Modbus Slave

The first line of the file contains the following information:

`ModbusSlave;;;`

- **Info1**: Not used.
- **Info2**: Address and size in bytes of the register in the `<adresse>_<taille>` format. If the size is 1 register (2 bytes), only `<adresse>` is required. The size in bytes must be a multiple of 2.
- **Info3** Not used.
- **Info4** Name of the device targeted by this association, that is, either:
  - registered equipment
  - a Lua script

- **Name**: not used.
- **Tag**: Tag of the variable targeted by this association. This field must not be used if the Name field is used.
- **CoefA**: Not used.
- **CoefB**: Not used.
- **Unit**: Not used.
- **Action**: Contains the code describing the types of possible actions, which are:
  - **0**: variable disabled. The variable is not used.
  - **1** (default value): The target variable is read-only. Write access is not permitted.
  - **4**: The targeted variable is readable and writable.

> **Warning**
>
> Variables are referenced either by name (**name**) or by their tag (**tag**)

Exemple de fichier de définition

In the example below, we retrieve variables from two different inverters and one variable created by a Lua script, as shown in the following list:

1. Read the variable named **Temperature** in U16 format from **INVERTER1** (1 register, or 2 bytes).
2. Read the variable named **E-Total** in F32 format from **INVERTER1** (2 registers, or 4 bytes).
3. Read the variable named **Firmware Version** as a string with a maximum length of 20 bytes from **INVERTER2** (i.e., 10 registers).
4. Read the variable tagged **DisplaySerialNumber** in String format, with a maximum length of 24 bytes, from INVERTER2 (i.e., 12 registers).
5. Read and write the **ValueRegulation** variable of the virtual device in F32 format, created by the Lua script **regulation.lua** (4 registers, or 8 bytes).

```
ModbusSlave
1;;0;;INVERTER1;Temperature;;;;;1
2;;1_4;;INVERTER1;E*Total;;;;;1
3;;3_20;;INVERTER2;Firmware Version;;;;;1
4;;13_24;;INVERTER2;;DisplaySerialNumber;;;;1
5;;25_8;;regulation;ValueRegulation;;;;;4
```

Proprietary protocols

- **Info1**: See the corresponding appendix
- **Info2**: See the corresponding appendix
- **Info3**: See the corresponding appendix
- **Info4**: See the corresponding appendix

2.4 Field Communications Manager

2.4.1 General Principle

The field communication manager handles communication between the gateway and field devices using supported industrial protocols.

The gateway acts as the master for all configured buses. Field devices are actively queried using a polling mechanism.

The following principles apply:

- All communications are initiated by the gateway
- No field equipment can initiate unsolicited communication
- Communication takes place using a request/response model
- The collected data is made available to other modules in the system

The field communication manager does not perform any business-specific data processing. It is responsible solely for acquiring and transmitting the data.

2.4.2 Supported Protocols

Many devices can be connected to the product using various protocols:

Protocol	Interface	Equipment by Interface	Monitoring	Control
Modbus RTU	serial 1/2/3	247 max	✓	✓
Modbus TCP	lan 1/2	247 max	✓	✓
SunSpec Modbus RTU	serial 1/2/3	247 max	✓	✓
SunSpec Modbus TCP	lan 1/2	247 max	✓	✓
SMA-net	serial 1/2/3	100 max	✓	✗
Solarmax	serial 1/2/3	100 max	✓	✗
PowerOne	serial 1/2/3		✓	✗
Delta-Solivia	serial 1/2/3	100 max	✓	✗
Kaco RTU	serial 1/2/3	32 max	✓	✗
Kacto TCP	lan 1/2	100 max	✓	✗
Diehl Platinum	serial 1/2/3		✓	✗
Siemens Refusol	serial 1/2/3		✓	✗

It is also possible to connect devices to the other interfaces

Protocol	Interface	Equipment by Interface	Monitoring	Control
ICT	usb	1 max	✓	✗
0–10 V	analog 1/2/3/4	1 max	✓	✗
4–20 mA	analog 1/2/3/4	1 max	✓	✗
Pulsed S0	digital 1/2/3	1 max	✓	✗
dry contact	digital 1/2/3	1 max	✓	✗

2.4.3 Bus Management

The gateway can handle multiple buses simultaneously.

Each bus:

- is configured independently
- has its own settings
- is associated with a specific set of equipment

The buses are logically isolated:

- A fault on one bus does not affect the other buses
- Communication cycles are specific to each bus

> **Warning**
>
> A bus can support only one protocol at a time

Each piece of equipment is associated with a single bus.

A bus can connect multiple devices.

This association determines:

- the communication channel used
- the settings applied to the equipment

The device addressing information is defined in Device Manager and used by the field communication manager to establish communications.

2.4.4 Principle of Communication

The gateway acts as the master and initiates all communication with the field devices.

Communications are organized using a continuous polling mechanism.

Polling

The gateway actively polls the devices. Polling is performed in a continuous loop. Each exchange follows a request/response pattern.

On a given bus:

- The equipment is processed sequentially
- For each device, Modbus requests are executed sequentially in the order in which they are defined.

Spontaneous communications from the devices to the gateway are not taken into account in the polling process.

A cycle corresponds to a complete iteration through all the devices configured on a bus.

For each piece of equipment:

- All defined queries are executed in order
- The acquired data is organized into a dataset
- An aggregated communication status is associated with the equipment
- The polling then moves on to the next device

The duration of a cycle depends, among other things, on:

- the number of pieces of equipment
- the number of requests per device
- response times
- configured timeouts
- any communication errors
- external requests injected into the bus

Handling Request Failures

When a request fails:

- No variables are generated for this query
- Variables successfully acquired earlier in the equipment cycle can be retained in the dataset published at the end of the cycle

> **Note**
>
> A piece of equipment's dataset may therefore be incomplete.

> **Note**
>
> In the event of an access conflict requiring reprocessing, the device may be skipped for the current cycle. In this case, the partial variables are not published.

Multi-bus management

The gateway supports several independent communication buses:

- up to three configurable serial buses
- a Modbus TCP bus

Each bus has its own communication engine.

> **Note**
>
> The buses operate independently of one another. A failure on one bus does not interrupt the operation of the other buses, although it may affect the system's overall performance.

Integration of External Requests

Some functions can inject external requests into the communication stream, particularly through a mechanism that converts Modbus TCP to Modbus RTU.

These requests are processed by the same communication engine as internal polling.

Data Availability

The acquired data is made available to the other modules in the system once each piece of equipment has been processed.

The timeliness of the data depends directly on:

- the bus cycle time
- the number of requests
- communication errors
- the load caused by external requests

2.4.5 Error Handling

The field communication manager handles communication errors at the level of each request and each piece of equipment.

Errors are detected while queries are being executed and affect the polling process as well as the status of the devices.

Types of Errors

The following errors may occur:

- **TIMEOUT**: No response within the specified time frame
- **CRC**: Integrity error in the frame (RTU)
- **EXCEPTION_RESPONSE**: Modbus exception response
- **UNEXPECTED_FRAME**: Unexpected or invalid frame
- **TCP_CONNECTION_FAILURE**: Failed to establish a TCP connection
- **SOCKET_CLOSED**: TCP connection closed during data exchange
- **CONGESTION**: System overload preventing requests from being processed in a timely manner
- **OTHER_ERROR**: Generic error

Behavior in the Event of a Request Error

When an error occurs in a request:

- The request is considered to have failed
- No data was returned for this query
- Polling continues according to the following rules

General case:

- The system moves on to the next request from the device

In the event of a timeout or loss of connection:

- The remaining requests from the current device are ignored
- The polling process immediately moves on to the next device

Impact on Data

When one or more requests fail for a piece of equipment:

- Data associated with failed queries is not updated
- Data successfully acquired during the same cycle can be retained

A piece of equipment's dataset may therefore be incomplete.

> **Note**
>
> Variables that are not updated retain their last known value. They are neither deleted nor reset.

> **Note**
>
> The exact behavior of unupdated values (retaining the last value or showing no data) depends on how the higher-level modules handle the data.

Equipment Communication Status

After a piece of equipment has been processed:

- A communication status is associated with the equipment
- This status reflects the overall result of the queries that were executed

Communication errors are first mapped as follows:

- **OK** → Status: OK
- **TIMEOUT**, **TCP_CONNECTION_FAILURE**, **SOCKET_CLOSED** → status: NOK
- **CRC**, **EXCEPTION_RESPONSE**, **UNEXPECTED_FRAME**, **CONGESTION**, **OTHER_ERROR** → status: ERRORS

The final status of the device is determined based on a history of the same size as the number of requests for that device:

- if all requests result in a critical failure → NOK
- if at least one error of type ERRORS is present → NOK
- if at least one query is OK and there are no errors, ERRORS → OK
- otherwise → NOK

2.5 Remote Connection Manager

2.5.1 Role

Remote Connection Manager provides remote connectivity for the gateway. It manages connectivity for **LAN** and **mobile** to enable IP communications for the product.

The Remote Connection Manager covers:

- the configuration and implementation of **Ethernet** connectivity
- **Cellular** Connectivity Management
- the exposure of the network status to other system modules
- managing the DNS resolution and routing settings required for remote connectivity

The module provides a **network transport** function. It does not handle the business logic of application exchanges.

2.5.2 Ethernet

The module supports up to **2 LANs**. For each interface, it allows you to define and apply an IPv4 configuration that includes, among other things:

- IP address
- subnet mask
- walkway
- Primary DNS
- Secondary DNS

Associated Status

The module exposes functional states that can be used by other components and by diagnostic functions:

- configured / not configured
- online / offline
- gateway reachable / unreachable

2.5.3 Mobile

The module manages mobile connectivity through the built-in modem.
This includes:

- Modem Initialization
- registration on the operator's network
- Establishing a data session
- Whether the connection is maintained temporarily or permanently, depending on the configuration
- closing the session when it is no longer needed

Associated Status

The module exposes functional states that can be used by other components and by diagnostic functions:

- network registration status
- configured / not configured
- online / offline
- network type (2G/4G)
- operator
- radio quality (RSSI, CSQ, dBm)
- session IP settings

2.5.4 Behavior

The Remote Connection Manager applies the following rules:

- Any change to the LAN configuration results in a consistent reapplication of the network configuration
- A LAN configuration is validated before implementation
- IPv4 settings must be consistent with one another
- The IP addresses configured on the LAN interfaces must be unique
- Mobile connectivity can be established on demand or according to a configured maintenance policy
- If there is no active request, a mobile session may be temporarily retained before being closed
- In the event of a mobile failure, the module enforces a recovery policy
- The state exposed to the other modules corresponds to the connectivity's status, not just the requested configuration

2.5.5 Global Connectivity

The remote connection manager operates in two parallel subdomains:

- an area **Ethernet**, focused on the validation and implementation of a local configuration
- an area **mobile**, focused on establishing, maintaining, and reestablishing a data session

The system is considered **connectable** as long as at least one network path—Ethernet or mobile—is operational.

2.5.6 Selecting the Communication Path

The network manager does not choose between the various available connectivity paths.

It displays the status of the **Ethernet** and **mobile** connections, but does not determine the path used for communications.

The choice of connection type (LAN or mobile) is made by the client modules—specifically, the server manager—based on their configuration.

The network administrator is responsible only for ensuring the availability and condition of the interfaces, regardless of how they are used.

2.6 Remote Server Manager

2.6.1 Role

The **remote server manager** allows you to perform configuration and management of communication between the gateway and external servers.

He says:

- sending data (measurements, alarms, logs)
- retrieving commands and configuration files

Each server has its own configuration and operates according to its own settings.

2.6.2 Server Configuration

The system supports **two configurable servers**:

- **Server 1**: dedicated to file transfers
- **Server 2**: supports file transfers or MQTT

> **Note**
>
> Server 1 is the main server that allows you to modify the configuration remotely.

> **Note**
>
> (Only Server 2 can use MQTT)

For each server, the user can configure:

- server type (FTP, SFTP, WebDAV, MQTT, etc.)
- address and port
- login credentials
- network connection type (LAN or mobile)

> **Note**
>
> A server is considered active once its configuration is valid.

Specific Settings

Depending on the type of server:

- remote directories (files)
- Topics and QoS (MQTT)

- certificates and security settings

2.6.3 Planning

Exchanges can be initiated:

- automatically according to a schedule
- manually by the user (via the web interface or SMS)
- following certain events (e.g., alarms)

Each server handles its own transactions independently:

- A server can execute only one transaction at a time
- Subsequent requests are placed on hold

Inter-server communication depends on the transport type:

- Two file-mode servers do not run in parallel
- A file server and an MQTT server can run in parallel

In the event of a conflict, an exchange may be delayed or fail.

> **Note**
>
> The first connection occurs no sooner than 5 minutes after the product's startup.

2.6.4 What to Do in Case of Failure

If the connection or exchange fails:

- Each execution performs up to **2 immediate tests**
- If a failure persists during a scheduled run, up to **3 new runs** are automatically retried
- These new executions are spaced approximately **1 minute** apart
- Each new execution performs up to **2 immediate attempts** again

After these attempts:

- No new tests will be conducted until the next scheduled activation

Manual actions do not trigger an automatic retry

2.6.5 Behavior Based on Network Connectivity

The server uses the configured connection mode (LAN or mobile).

- If mobile connectivity is not available: the system temporarily waits for a connection before failing
- If LAN connectivity is unavailable: The system attempts the transfer, but it fails if a connection cannot be established.

In some cases, an alternative connection can be used if the network configuration allows it.

2.6.6 MQTT Behavior

MQTT functionality depends on the configuration:

- **persistent connection mode**:
  - The connection to the broker is maintained continuously

- Automatic reconnection in case of loss

- **standard mode**:
  - A connection is opened for each exchange and then closed

Transactions follow the server's execution cycle (scheduled or triggered).

2.6.7 Oversight and Bylaws

For each server, the user can view:

- Status: In Progress / Inactive
- latest result:
  - success
  - failure
  - canceled

These statuses reflect the overall result of the last execution.

2.7 System Manager

2.7.1 Firmware

The firmware consists of all the embedded software components of the gateway.

It is identified by a software version number and can be updated to provide fixes, improvements, or new features.

Software Version

The gateway displays the following information:

- application firmware version
- kernel version
- product model
- serial number
- date of last update

This information is available in whole or in part on the web server, as well as via external commands (SMS/ Modbus Slave, MQTT, or SFTP).

Firmware Update

The gateway supports two update modes:

- Local update via a user interface
- remote update via a server.

In both cases:

- The firmware is provided as a package **spm**
- The package's integrity is verified before installation
- An invalid update is rejected

> **Note**

SPM packages are encrypted to prevent tampering

Installation Process

The update process involves the following steps:

1. Receipt of the firmware package
2. integrity check
3. preparation for the new system
4. gateway restart
5. Activation of the new version

A system restart is required to complete the installation.

Behavior After the Update

After a firmware update:

- The gateway restarts automatically
- Persistent configuration settings are retained
- User files are retained
- Installed licenses are retained

Catering

The gateway incorporates several automatic restoration and recovery mechanisms designed to ensure its ability to start up and operate.

Automatic Configuration Restoration

At startup, the system can automatically restore a local configuration backup if one is available.

This restoration occurs before the configuration is loaded normally.

It allows you to restore a previous operating state in the event of an update or change that caused a malfunction.

Start-up Validation

After restoration, the system verifies that the system has started up correctly.

As long as the startup is not considered stable, the configuration backup is retained.

Once the startup is confirmed, the backup is automatically deleted.

Recovery After Abnormal Restarts

The system detects repeated abnormal reboots (reboot loop).

In this case, a recovery mode is automatically activated.

This mode involves:

- gradually isolate the configuration elements that may be causing the malfunction,
- try to restart with a partial configuration,
- restore minimal functionality if necessary.

This mechanism may result in a partial or total loss of the local configuration in order to ensure that the system restarts.

Factory Reset

The gateway supports a full reset (factory reset).

This operation will delete the local data and restore the system to its initial state.

It is distinct from automatic recovery mechanisms.

Limitations

- There is no automatic update process; updates are always initiated by the user.
- Downgrading is not recommended

2.7.2 Local System Settings

The gateway has local system settings used for identification, time management, and the operation of certain cross-product functions.

These parameters are distinct:

- network configuration
- configuration of field equipment
- configuring communication with external servers

They are stored locally and reused upon restart, depending on their type.

Date and Time

The gateway uses the local system time to time-stamp its logs, alarms, files, and exported data.

The system time can be updated:

- automatically via NTP synchronization
- manually by user action

After synchronization or manual adjustment, the system time is copied to the hardware clock so that it is retained between reboots.

Upon startup, the system can retrieve the time from the hardware clock if it is considered valid.

If no valid time is available at startup, the behavior depends on the platform's state until NTP synchronization or manual time setting is performed.

The system time is used, in particular, for:

- system and application logs
- alarms
- files named with the date and time
- the exported data
- the display of the current time

Time zone

The gateway has a configurable time zone.

The time zone is used for local formatting of dates and times in the various system components.

In particular, it affects:

- the display of the current time
- newspapers
- alarms
- file names with dates
- data exports
- schedules that use local time
- scripts that use the system's local time

If no valid time zone is specified, the system uses UTC as the default reference.

Product ID

The gateway displays various system identifiers, including:

- a gateway ID
- a serial number
- a product model
- the firmware version
- the kernel version
- the software agent version

The gateway ID is a configurable parameter. It is used by the product to identify the device in certain interfaces and in various file-naming and exchange mechanisms.

The serial number and software versions are read-only system information.

Persistent settings

The gateway stores a set of persistent system settings locally.

These parameters include, in particular:

- the gateway ID
- NTP settings
- the time zone
- certain access settings
- certain SMS settings
- certain logging settings
- certain system settings related to software behavior

System settings are saved locally and reloaded at startup.

The current time is not stored as a configuration parameter. Its persistence across reboots relies on the hardware clock and time synchronization mechanisms.

The system distinguishes between several persistence domains, including:

- local system settings
- scheduling parameters
- equipment settings
- data and logs
- network settings

Admission Rules

The parameters that the user can enter must conform to the following list:

Appetizers	Accepted characters	Terms and Conditions
HMI Login	UTF-8: displayable	max:255
HMI Password	UTF-8: displayable	min:4 max:128
SMS Password	UTF-8: displayable	max:128
Server User	UTF-8: displayable	max:128
Server password	UTF-8: displayable	max:128
equipment name	a-z A-Z 0-9 / _ - [ ] <space> ( ) .	
tag	a-z A-Z 0-9 / _ - [ ] <space> ( ) .	
category	a-z A-Z 0-9 / _ - [ ] <space> ( ) .	
script name	a-z A-Z 0-9 / _ - [ ] <space> ( ) .	
gateway ID	a-z A-Z 0-9 / _ - [ ] <space> ( ) .	
server name	IP Address / Domains	
server directory name	a-z A-Z 0-9 / _ - .	
MQTT topic	a-z A-Z 0-9 / _ - .	
variable name	a-z A-Z 0-9 / _ - [ ] <space> ( ) : .	except , ; "
variable tag	a-z A-Z 0-9 / _ - [ ] <space> ( ) .	
unit	UTF-8: displayable	

2.7.3 System Maintenance

The gateway includes maintenance features that allow you to restart the system, reset the device, and ensure automatic recovery in the event of a failure.

These mechanisms are designed to ensure the product's operational continuity and its ability to restart.

Restart

The gateway supports two levels of reboot:

- **Software reboot**: Restarting software services without a full system reboot
- **System reboot**: Full gateway restart

A system restart can be initiated:

- from the user interface
- via a remote command
- or by the system itself in some cases

A system restart results in:

- the suspension of current services
- a complete restart of the platform
- the preservation of persistent settings and data

Application restart is primarily used as an internal maintenance and recovery mechanism.

Some remote commands may be executed later to allow ongoing transactions to complete properly.

Return to Factory

The gateway supports a full reset (factory reset).

This operation:

- is triggered by a remote command or a physical action
- is executed on the next restart (not immediately)
- results in the deletion of local application data

A factory reset involves, among other things, deleting:

- of the local configuration
- newspapers
- acquisition data
- scripts and related data

The firmware and hardware identifiers are not affected.

After a reset, the product restarts in its initial state, with the minimum structures necessary for its operation being recreated.

Automatic Recovery

The system includes automatic recovery mechanisms designed to ensure its ability to start up.

In the event of repeated abnormal restarts, the system:

- detects a reboot loop
- attempts to isolate the faulty configuration elements
- can gradually remove certain configuration files
- can revert to a minimal configuration if necessary

The goal is to restore the system to normal operation.

This mechanism may result in a partial or total loss of the local configuration.

Diagnostics and System Status

The gateway displays system information that enables diagnosis of its status, including:

- firmware version
- kernel version
- product ID
- serial number
- model
- date of last update

The system also provides:

- An Overview of Communication Services
- system and application logs
- diagnostic export mechanisms

A diagnostic bundle can be generated to compile the information needed for support (logs, traces, etc.).

Maintenance events (restarts, resets, recoveries) are recorded in the system logs.

2.7.4 Logs

Log files are files that allow you to monitor and analyze the gateway's activities.

There are 5 types of log files:

- Log Files
- LUA Script Execution Logs
- SunSpec Detection Logs
- System Logs
- Communication Diagnostic Logs

Here is a summary:

Journal	Web Server Visibility	Transfer to Server 1/2	Persistence	Export
Log In	✓	✓	✓	✗
LUA Script	✓	✓	✓	✗
SunSpec Detection	✓	✓	✓	✗
Systems	✗	✓	✓	✓
Communication Diagnostics	✓	✓	✓	✓

Note

The export feature is available from the web server to generate an archive for use by Webdyn Support when analyzing logs.

2.7.5 Licenses

Licenses enable certain features of the gateway, including control scripts and energy management.

A license is specific to an identified product and cannot be used on another gateway.

License Status

The system considers the following statuses:

- **valid**: The license is installed and enables the use of the associated features
- **missing**: No license installed
- **invalid**: License present but not recognized or not applicable to the product

Obtaining a License

The license is obtained from the sales department. Each license is specific to a gateway: the gateway's unique identifier is used to generate the license. It is therefore important to provide this identifier when placing an order.

The unique identifier `UID` is: **WPMxxxxxx**, with **xxxxxx** representing the last 6 digits of the gateway's serial number or MAC address.

> **Note**
>
> By default, the gateway's name matches its `UID` and is visible on the web server (in the banner, the **System>About** tab, or in the `<UID>_CONFIG.ini` file).

Installation and Updates

A license can be installed or replaced at any time during the product's lifecycle.

This can be installed by uploading a file to Server 1 or directly from the gateway's web server.

Persistence

Installed licenses are perpetual and are retained when:

- gateway reboot
- gateway update

There is no mechanism for automatically transferring a license between devices. The license must be renewed when the gateway is replaced.

2.8 Modbus TCP Slave

The gateway features a Modbus TCP slave function that allows it to be integrated into a system with a central control or communication unit, such as a SCADA or BMS system, by providing a flexible communication interface.

This feature allows you to:

- to access certain information specific to the gateway (serial number, firmware version, etc.),
- to access the variables of the devices managed by the gateway,
- to access the variables of the virtual devices created by the scripts,
- to execute commands from the gateway.

2.8.1 General Operation

The Modbus slave is accessible only via TCP at the slave address **1** and is available on both Ethernet interfaces (LAN1 and LAN2). All Modbus registers are in the 16-bit format **holding registers**. The Modbus slave registers are divided into two parts, which are:

- **User defined registers**: These registers are addressable from 0x0000 to 0x7FFF (0 to 32,767). They can be used freely and configured via a definition file.
- **Webdyn defined registers**: These registers are addressed in the range from 0x8000 to 0xFFFF (32768 to 65535). They are reserved and cannot be configured. They provide access to gateway information.

Each register can be read using the following Modbus functions:

- **0x03** (Read Holding Registers): read
- **0x06** (Write Single Register): single write
- **0x10** (Write Multiple Registers): multiple writes

A variable can be associated with a group of one or more adjacent registers.

When a variable is accessed, the behavior is as follows:

- **Reading**: The gateway returns the last value read for this variable. This can be:
  - A message from the gateway
  - A variable associated with a connected physical device (for example: an inverter, an energy meter, etc.).
  - A variable of a virtual device created by a Lua script during runtime.
- **Writing**: The gateway changes the value of the variable. This may be:
  - The gateway's relay output. This allows you to change the state of the relay output.
  - Sending an order.
  - A variable associated with a connected physical device (for example: an inverter, an energy meter, etc.). This action requires a request to be sent to the physical device.
  - A variable of a virtual device created by a Lua script during runtime.

2.8.2 Read or write a variable

When a group of registers associated with a variable is read, the last known value of that variable is returned in a standard Modbus response.

When a group of registers associated with a variable is written, the concentrator writes the new value to the device as soon as possible. The response is returned to the client only after this operation is complete.

2.8.3 Execute a command

To execute a command in Modbus slave mode, simply write a character string to register address 34000 using the same syntax as for an SMS command, with a maximum length of 120 characters.

2.8.4 Predefined Webdyn Variables

The predefined Webdyn variables are stored in the WebdynSunPM register range from 0x8000 to 0xFFFF (32768 to 65535). WebdynSunPM variables cannot be configured and provide access to certain information from the concentrator.

Webdyn variables are available at the following Modbus register addresses:

Address and number of registers	Data type	Access	Description
33,000 / 15	Thong	Reading	Firmware Version
33015 / 15	Thong	Reading	Kernel version
33030 / 15	Thong	Reading	Serial number
33045 / 15	Thong	Reading	Username
33060 / 2	"time_t in swap _W"	Reading	Date of last update
33,200 / 2	U32	Reading	Digital Input 1
33202 / 2	U32	Reading	Digital Input 2
33204 / 2	U32	Reading	Digital Input 3
33206 / 2	U32	Reading	Analog Input 1
33208 / 2	U32	Reading	Analog Input 2
33210 / 2	U32	Reading	Analog Input 3
33212 / 2	U32	Reading	Analog Input 4
33214 / 2	U32	Reading/Writing	Relay Output
34,000 / 120	Thong	Writing	Orders*

> **Note**
>
> **Commands**
>
> This registry group is special. It is not associated with a variable or any data, but it allows you to execute commands.

2.8.5 User variables

User variables are stored in the user register area, which is addressable from 0x0000 to 0x7FFF (0 to 32,767).

User registries are configured via a **definition file** located in the configured definition directory (by default, `/DEF`) on Server 1 (the primary server).

The definition file describes the mappings between user registers and device variables or variables in a Lua script, and is fully configurable by the user.

2.8.6 Modbus Error Handling

In the event of an error, the gateway returns a Modbus exception code. The following are the error cases:

- **0x01 - Illegal Function**: Unsupported function code—that is, a value other than 0x03, 0x06, or 0x10
- **0X02 - Illegal Data Address**: Consistency issue between the request and the accessed registers:
  - A registry is not configured.
  - A register is ignored (action code 0).

    - A register group is incomplete; that is, fewer registers are requested than the group contains.
    - Attempt to write to more than one variable.
    - Attempt to write to a read-only variable (action code 1).
    - Attempt to read the order log.
- **0x03 - Illegal Data Value**: The value entered is invalid
- **0x04 - Server Device Failure**: Invalid Modbus address
- **0x0A - Gateway Path Unavailable**: During a write operation, unable to connect to a device via TCP. This may be due to a network configuration issue.
- **0x0B - Gateway Target Device Failed to Respond**: When writing:
    - A device did not respond (timeout).
    - A device responded with an invalid frame (CRC error, malformed, etc.).
    - A device unexpectedly closed its TCP socket.
- **0x70 - Invalid Mapping**: A variable has a size that is incompatible with the number of registers read or written
- **0x71 - No Such a Variable**: The gateway does not recognize the device or variable
- **0x72 - Invalid Action**: Writing to the target variable is not supported (Input Register, U8, etc.).
- **> 0x80**: During a write operation, if a device responds with a Modbus exception, the exception is forwarded to the client by setting the most significant bit to 1—that is, by performing the transformation 0x80|code

2.9 Web Services

When enabled, Web Services are triggered by the following actions:

- Product Boot
- Uploading Data Files to the FTP Server
- Changing the configuration (loading or saving a configuration)
- Running a batch file
- Loading New Firmware

Web service calls are made using the following format:

```
<WebServices_URL>/confirm.php;NSITE=<uid>&ACTION=<action>&ACTION-COMP=<complément>
```

The `confirm.php` page is called at the URL <WebServices_URL> configured in the `<UID>_config.ini` file in the configuration directory (by default, `/CONFIG`). This page will receive various pieces of information about the actions and must process them. The information is as follows:

- **NSITE**: hub ID. This is the ID that was configured during commissioning
- **ACTION**: ID of the action that triggered the web service call. The list of actions is described below
- **ACTION-COMP**: For certain actions, additional information is provided. The complete list is described below.
- **RC**: is always 0
- **RC-COMP**: always empty

The following table lists the `ACTION` and `ACTION-COMP` fields:

Action	Web Services
First FTP connection since the hub was started	&ACTION=**VERSION**&ACTION-COMP=**<FW>**
Sending data from inverters or I/O devices to the main server	&ACTION=**UPLOADDATA**
Sending Alerts to the FTP Server	&ACTION=**UPLOADALARM**
Sending the gateway configuration files to the FTP server	&ACTION=**UPLOADGLOBAL**&ACTION-COMP=**<UID>_config.ini;<UID>_var.ini;<UID>_daq.csv**
Uploading FTP server configuration files to the gateway	&ACTION=**CONFIGGLOBAL**&ACTION-COMP=**<UID>_config.ini;<UID>_var.ini;<UID>_daq.csv**
Sending One or More Definition Files	&ACTION=UPLOADDEF&ACTION-COMP=**defFile1.csv;defFileN.csv**
Receiving one or more definition files	&ACTION=CONFIGDEF&ACTION-COMP=**defFile1.csv;defFileN.csv**
Running a batch file.	&ACTION=CONFIGBIN
Loading New Firmware	

The web service must return one of the following values:

Code	Description
00	OK
10	Unknown site ID
11	Unknown action code
12	Unknown RC received
13	MAC address missing
-1	Internal server error

3 Installation

3.1 Safety Instructions

Failure to follow these instructions may result in damage to the equipment and pose a danger to people.

3.1.1 Electrical Connection

> **Warning**
>
> All wiring work must be performed by a qualified electrician

The gateway is a Class III electrical device. It operates in the low-voltage safety circuit (TBTS) range (<50 V); power must be supplied via a safety transformer that provides reliable galvanic isolation between the primary and secondary windings.

> **Warning**
>
> The gateway may be damaged by electrostatic discharge (ESD)

3.1.2 Environment

The installation and environmental requirements are as follows:

- Protection rating: IP20
- Operating temperature: -5...40°C
- Relative humidity: 25–75% RH
- Storage temperature: -20...+85°C
- Pollution level: 2

It must be installed in an environment with the appropriate level of protection.

> **Warning**
>
> This equipment is not suitable for use in areas where children may be present

> **Warning**
>
> Do not install the equipment near a heat source

> **Warning**
>
> Do not install the equipment at a height greater than 2 meters.

3.1.3 Chemical agent

> **Warning**
>
> Do not use any solvents or chemicals to clean the concentrator.

3.2 Unboxing

3.2.1 Contents

The WebdynSunPM gateway comes with:

- An angled SMA antenna for the modem (taped to the back of the product)
- A battery (already installed in the product)

Warning

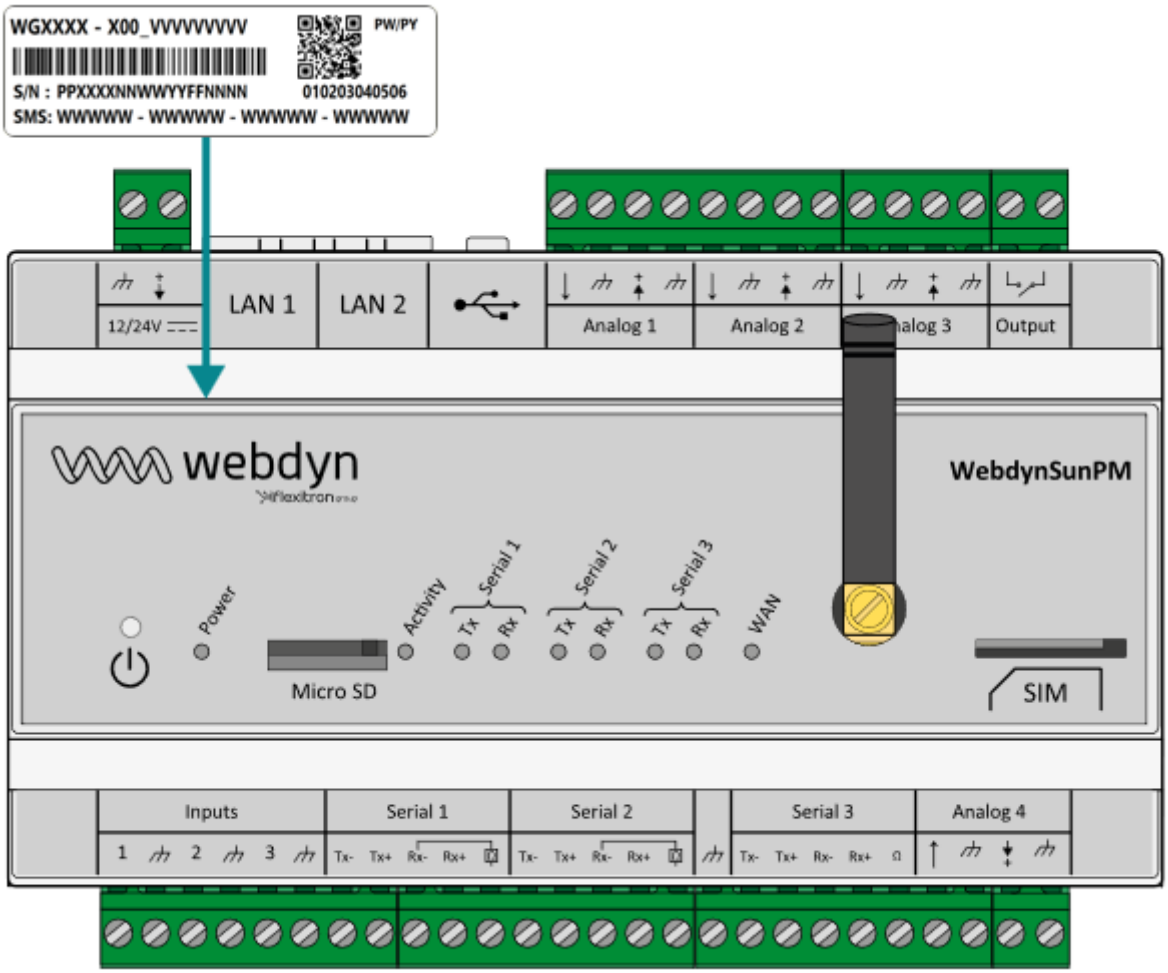
No SIM card is included with the gateway

Note

The power supply is sold separately

3.2.2 Identification

The product identification label is affixed to the gateway.



front with label

The information on the label is as follows:

sticker detail

- **Part number**: Sales Reference
- **PW/PY**: Date of manufacture in week/year format
- **S/N**: Serial number
- **Factory SMS Password**: Factory default password for SMS encryption
- **MAC Adress**: MAC address of the network card

The barcode contains the following information:

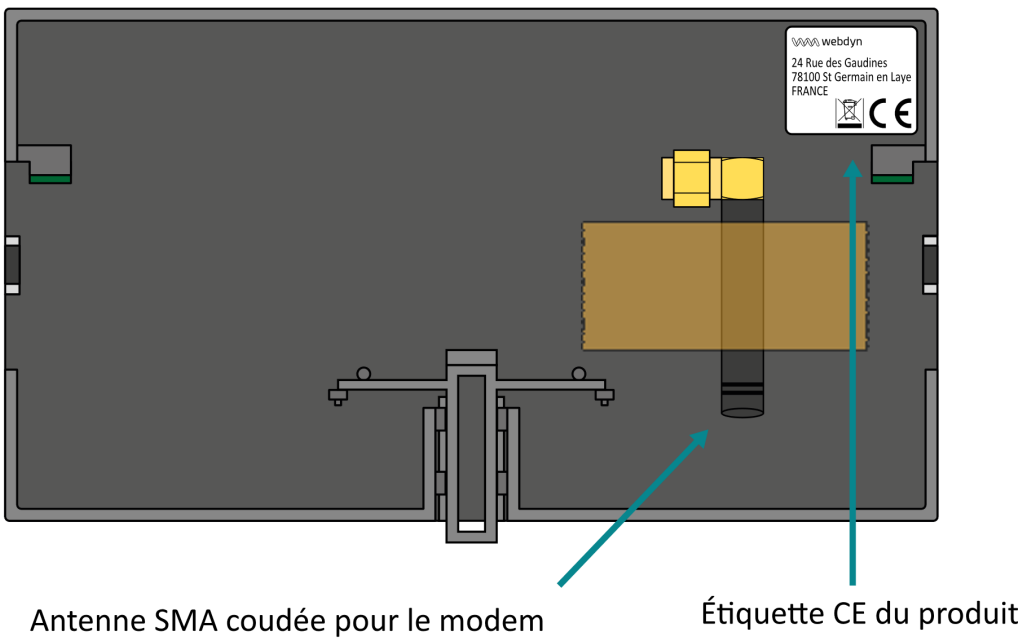
`S/N`

The QR code contains the following information:

`S/N;IMEI;SMS_factory_password`

Additional information can be found on the back of the gateway:

- Manufacturer's Name
- Manufacturer's Address
- Regulatory Markings



back label

3.3 Installation

3.3.1 DIN-rail mounting

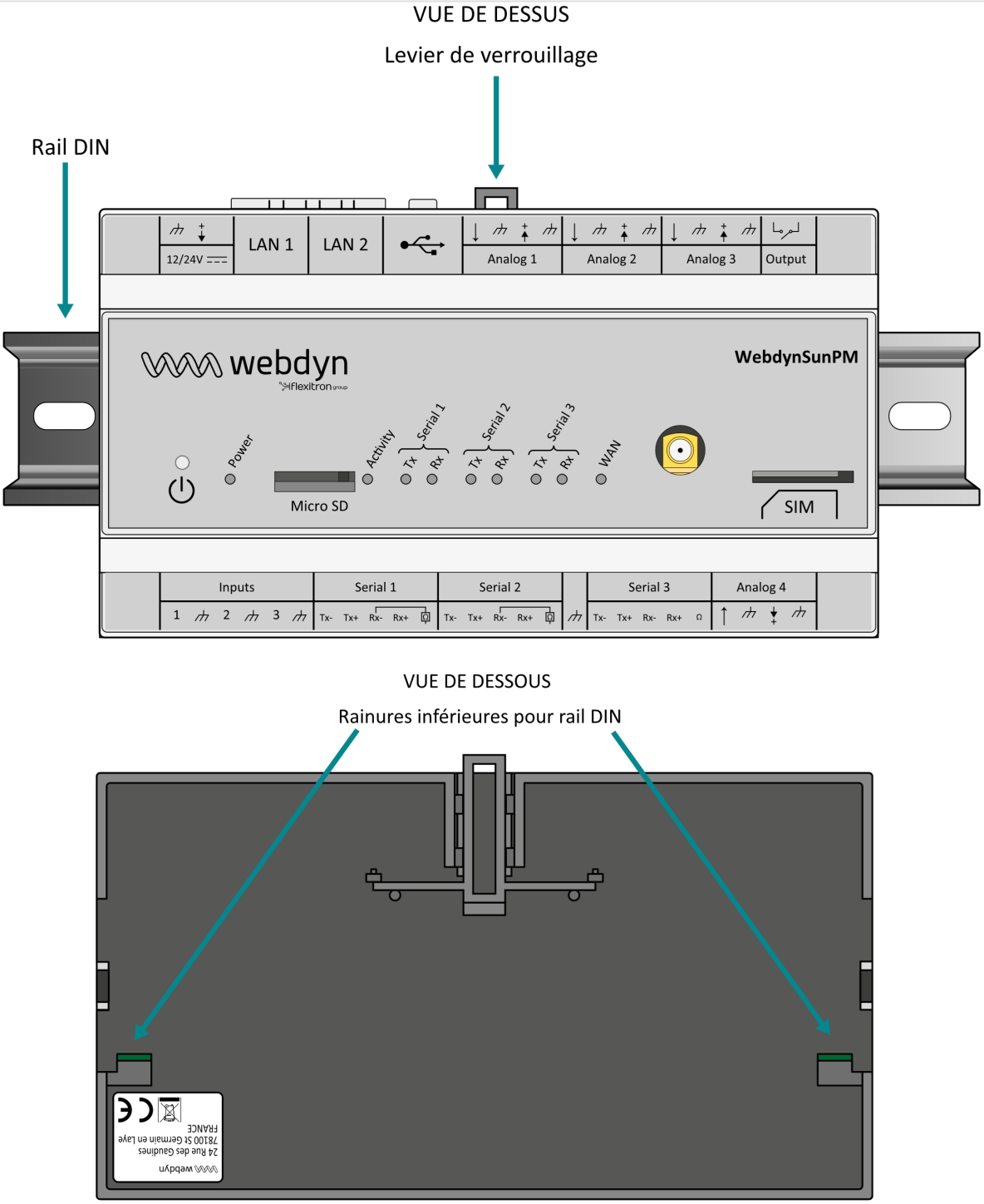
Warning

Before performing any assembly or disassembly work, make sure that:

- The gateway is not powered
- The antenna is removed from the back of the case

Warning

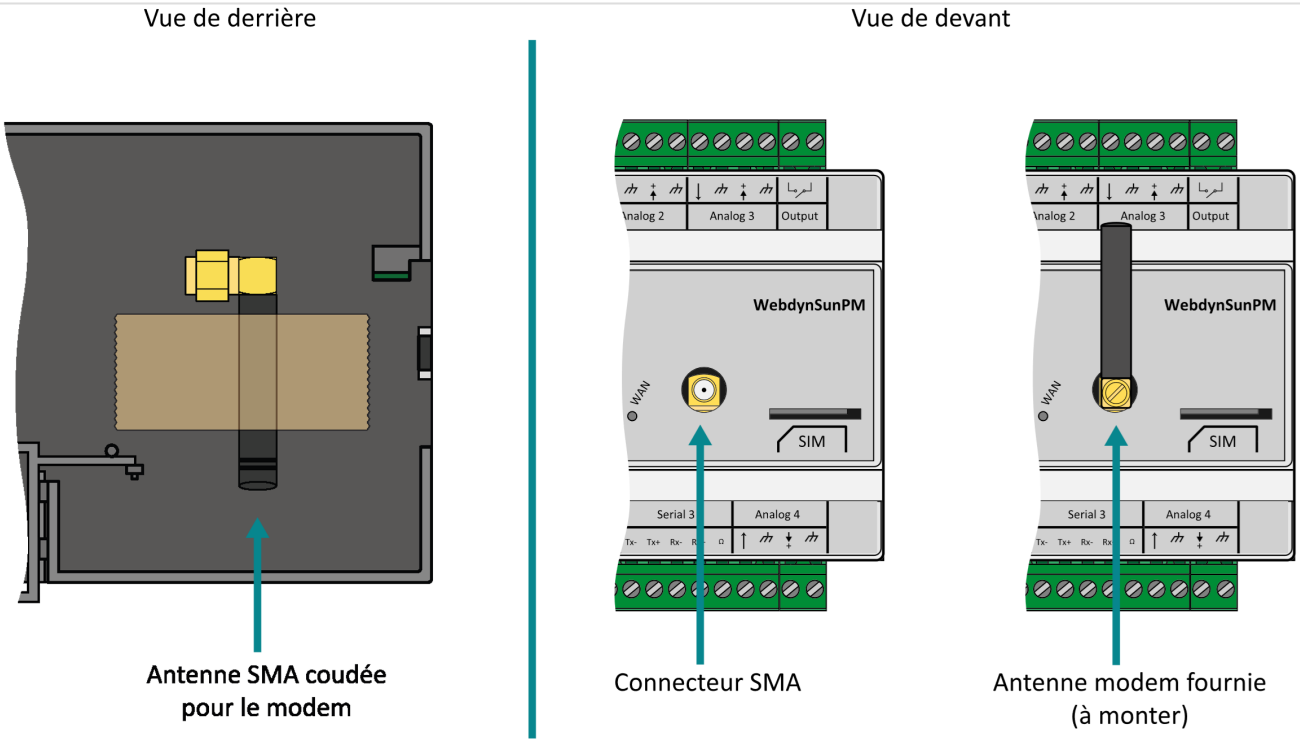
Opening the case is not permitted



The gateway is mounted on or removed from the DIN rail by operating the locking lever with a flathead screwdriver

3.3.2 Antenna

Proper antenna connection is essential to ensure the best possible cellular signal quality available at that location.



Antenna

Tip

To ensure good 4G signal reception:

- At least 20 cm of open space around the cabinet
- Do not place the gateway in a metal cabinet
- Do not place the gateway in the basement

If the product is installed in a metal enclosure or cage—a location that does not allow for proper signal reception—it is recommended to use a remote antenna

Warning

The remote antenna must be compatible with the connector and the frequencies supported by the gateway

Warning

The end user must ensure that their installation with remote antennas complies with current EMC standards.

Danger

An antenna must always be connected to prevent damage to the gateway.

3.3.3 SIM Card

The SIM card is inserted into the dedicated slot on the front of the device.

> **Warning**
>
> Please insert the SIM card in the correct orientation

3.3.4 Environment

The gateway is a device designed to be installed in an electrical enclosure that provides protection against the elements, splashes, and dust.

> **Tip**
>
> It is strongly recommended that the gateway be installed in an enclosure under a shelter that protects it from sunlight and backflow from equipment such as inverters.

3.4 Wiring

> **Danger**
>
> Before performing any wiring work, make sure the gateway and the electrical system are de-energized

3.4.1 Connectors

The gateway comes pre-installed with female cable connectors. These are screw-type cage terminals that require a flathead screwdriver for M3 screws.

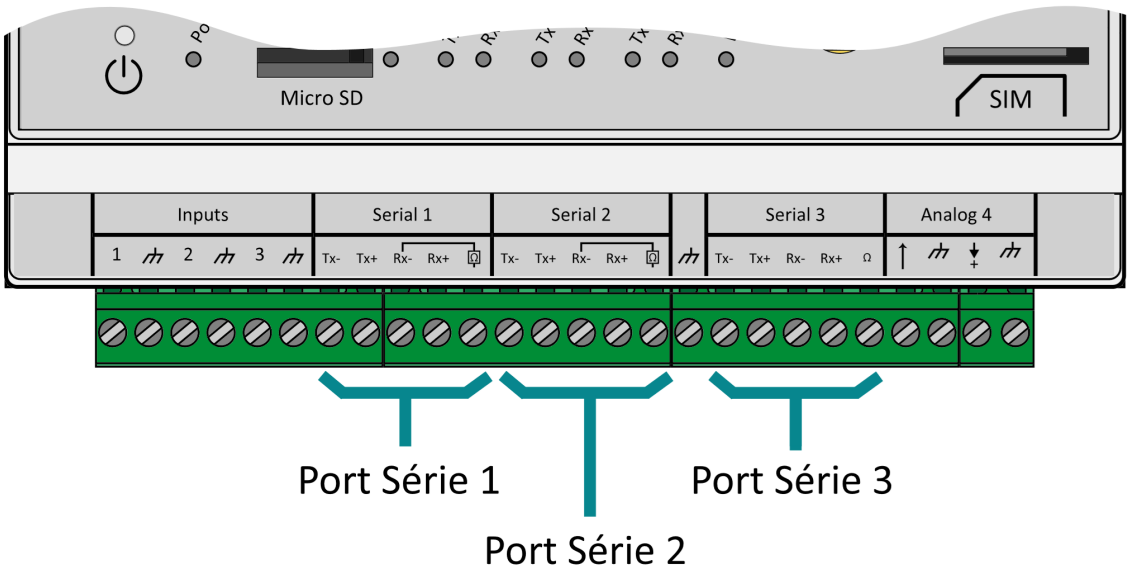
The features are listed below:

Feature	Specifications
Connector Type	Screw-Cage Terminal Block
Cross-sectional area for a single-strand cable	0.05...2.5 mm²
Cross-sectional area for multi-strand cable	0.05...1.5 mm²
Maximum/Recommended Torque	0.6/0.5 Nm
Stripping	7.5 mm

> **Note**
>
> If multiple cables need to be inserted into the cable cage, the use of ferrules is recommended

3.4.2 RS485/422 Serial Connection

The gateway is equipped with three RS485/422 serial ports. Each connection supports both half-duplex (2-wire) and full-duplex (4-wire) modes.



interface serials

> **Warning**
>
> Special care must be taken when wiring serial connections. Be sure to follow the recommendations and best practices for this type of connection regarding shielding, termination, and cabling topology. For more information on the RS485/RS422 standards and equipment cabling, please refer to the EIA-485 and EIA RS-422-A standards.

"Daisy-Chain" Topology

The serial connection is wired exclusively using **Daisy-Chain**. Each device is connected one after the other.

daisy chain wiring

> **Warning**
>
> Star wiring is prohibited

Cable Type

When choosing the type of cable, there are three distinct scenarios to consider, which are:

- For installations requiring short cable lengths and free from electrical interference, use a 2-pair 6/10 rigid shielded cable.
- For larger installations where the cable length does not exceed 500 m, use a 2-pair 8/10 rigid shielded cable.
- When the cable length exceeds 500 m—and especially in the event of electrical interference—use a shielded cable with two pairs of 0.34 mm² cross-section.

RS485 wiring on the gateway side:

- Strip about 4 cm of insulation from the RS485 communication cable.
- Trim the shielding down to the cable jacket.
- Strip the wires to a length of about 6 mm

120-ohm termination

To ensure proper operation of the data bus, an RS485 bus must be terminated at both ends with a 120-ohm *terminator*. The gateway can be installed either at the end or in the middle of the bus. The gateway includes a 120-ohm resistor that can be activated and is indicated on the connector with the **Ω** logo.

2-wire RS485 wiring (half-duplex)

This is the most common way to use the RS-485 standard. A single pair of wires is used for both transmitting and receiving data.

Connector	Description
Tx+	Differential Pair (+)
Tx-	Differential pair (-)
Rx+	Differential Pair (+)
Rx-	Differential pair (-)
GND	Common

Warning

Tx+ and Rx+ must be connected.

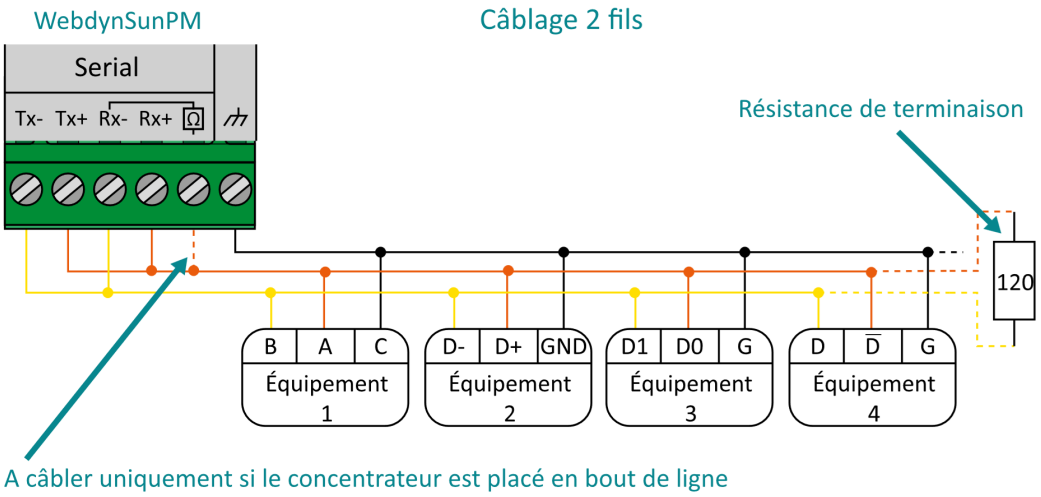
Tx and Rx must be connected.

Note

Make sure the **common** wire is connected to each piece of equipment. If it is not, the equipment must be properly grounded in accordance with the manufacturer's recommendations for each piece of network equipment. This requirement necessitates the use of an additional wire, which, although it does not participate in the communication process, is essential for ensuring the electrical integrity of the network devices.

Note

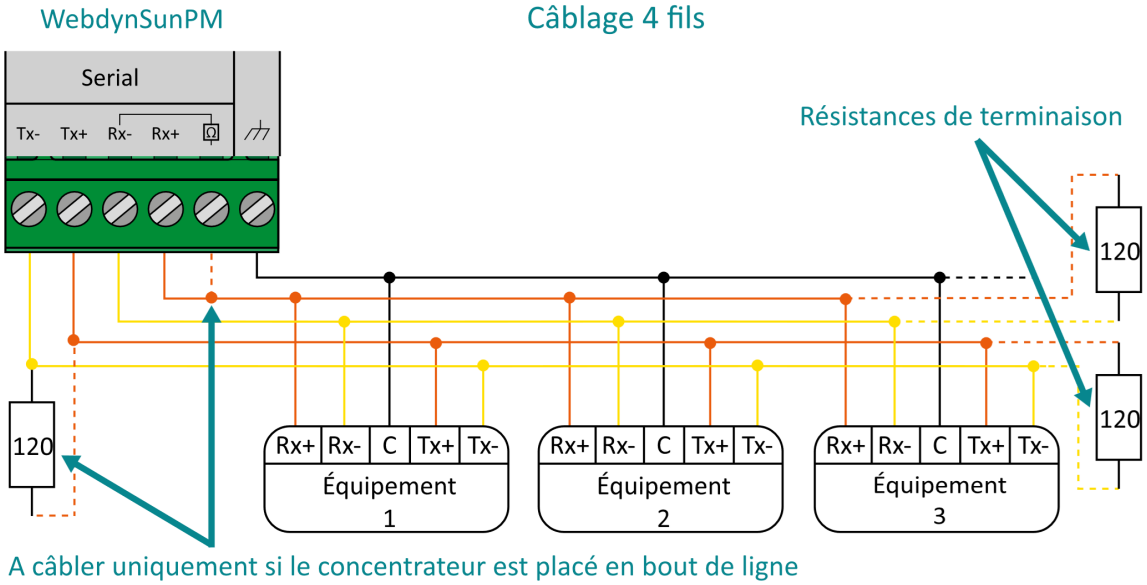
Different RS485 devices use different notations to indicate the correct connection configuration for differential communication pairs. The following figure shows some of the notations used.



serial 2wire wiring

4-wire RS485 wiring (full-duplex)

Two pairs of wires are used for communication. One pair of wires is used for transmission (gateway → devices), and the other pair is used for reception (devices → gateway).



serial 4wire wiring

Note

Make sure the **common** wire is connected to each piece of equipment. If it is not, the equipment must be properly grounded in accordance with the manufacturer's recommendations for each piece

of network equipment. This requirement necessitates the use of an additional wire, which, although it does not participate in the communication process, is essential for ensuring the electrical integrity of the network devices.

Shielding

The serial link must be shielded to eliminate electromagnetic interference and prevent coupling with the differential communication pair.

The shielding must:

- Be connected to the functional ground
- Be continuous along the entire length of the bus

To prevent the formation of **ground loop**, the shielding is usually connected to ground at **single point**.

> **Note**
>
> When the gateway is installed in the middle of the bus, be sure to maintain continuous shielding

Distance

The distance of the serial connection depends, in particular, on the baud rate used.

Baud rate (bps)	Typical Maximum Distance
9600	~1,200 m
19,200	~1,000 m
38,400	~800 m
57,600	~600 m
115,200	~400 m

Other factors also affect the maximum distance; it is recommended to reduce the communication speed when the distance is significant.

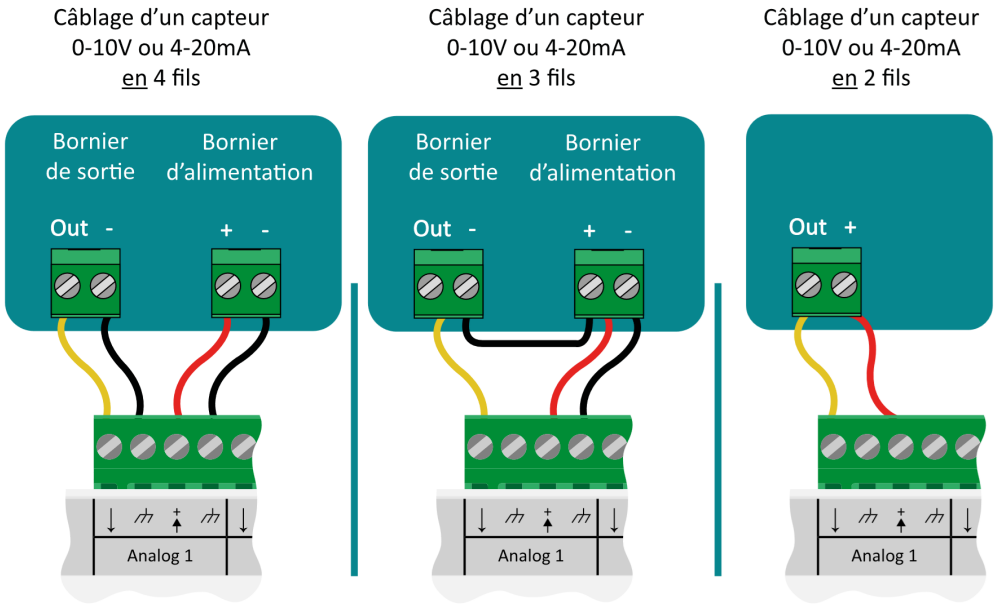
Maximum number of devices

The RS-485 standard specifies a maximum load of **32 Unit Loads (UL)** per bus segment. Each device connected to the bus has an input load expressed in UL (for example, **1 UL, 1/2 UL, 1/4 UL or 1/8 UL**, depending on the transceiver used). The sum of the loads of all connected devices must not exceed 32 UL. Although modern transceivers theoretically allow for the connection of more nodes, it is generally recommended to limit an RS-485 link to 32 devices to ensure reliable communication.

> **Note**
>
> The gateway's load is 1/8 UL

3.4.3 0-10V or 4-20mA analog inputs

Analog inputs can be 4-, 3-, or 2-wire. The following are the connection diagrams:



analog input wiring

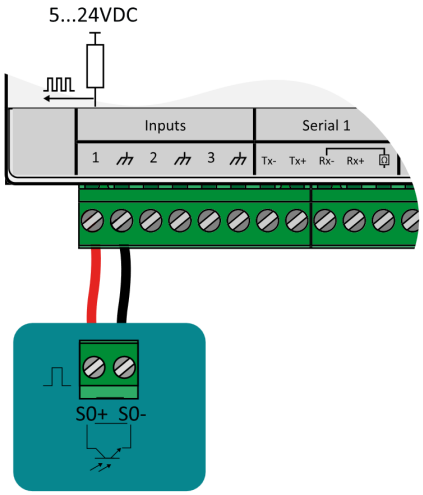
> **Note**
>
> The voltage available at the sensor terminal block is the same as that of the power supply unit used to power the gateway. The maximum power available for all sensors combined must not exceed 7 watts.

> **Danger**
>
> Do not apply a voltage greater than 12V or a current greater than 24mA

3.4.4 S0 pulse inputs or dry-contact inputs

Pulse Input Wiring

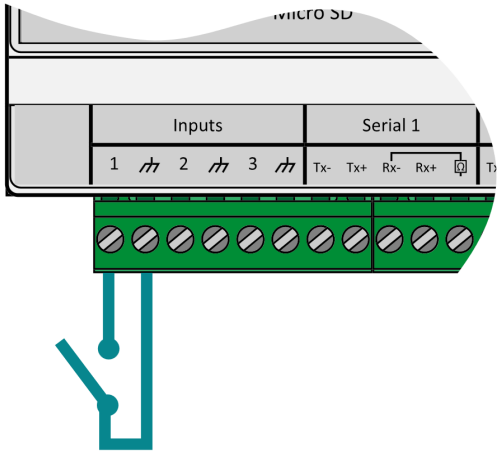
Câblage d’une entrée en S0



pulse input

Dry contact input Wiring

Câblage d’une entrée en TOR



dry contact

3.4.5 Relay Output

The relay output has the following characteristics:

Feature	Value
Maximum voltage	24V
Maximum current	1A

Warning

The relay must not be used to control heavy loads.

4 Commissioning

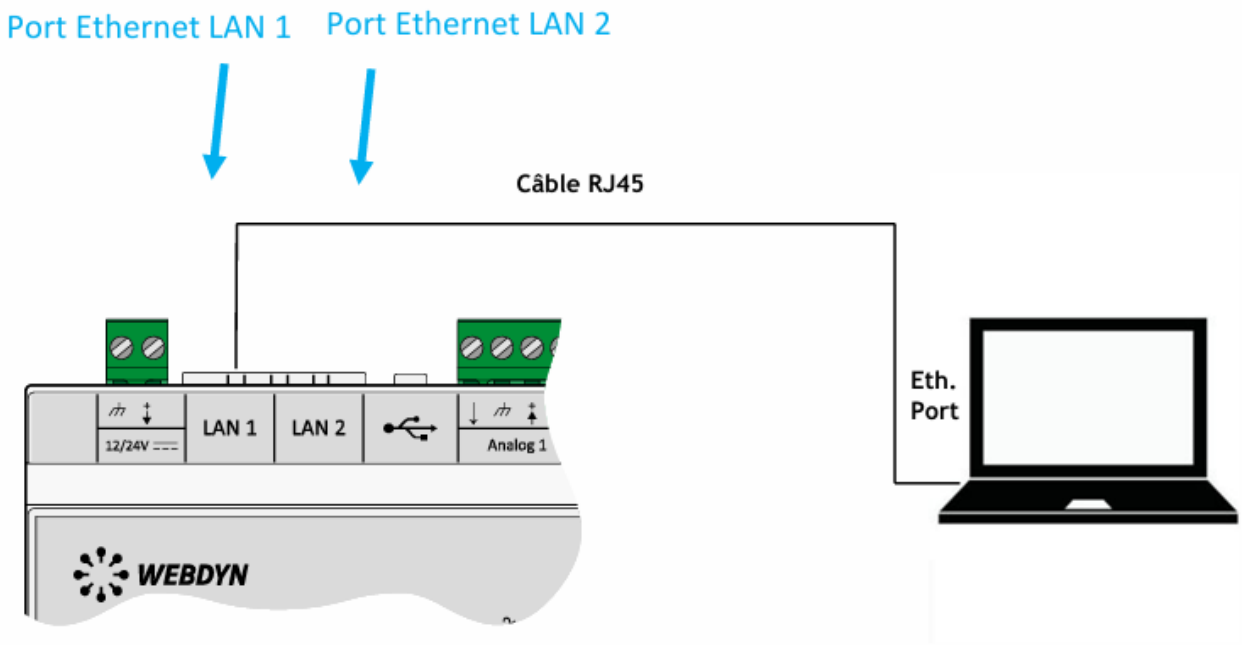
4.1 First Startup

The gateway is shipped with its factory settings, so you must configure it locally in order to operate it remotely.

4.1.1 Logging In to the Web Interface

First, you need to connect the WebdynSunPM to your computer's Ethernet port.

You can use the LAN1 or LAN2 port to connect to the web interface of the product.



pc lan connection

The gateway's web interface can be accessed at the following addresses:

- **LAN1:** https://192.168.1.12
- **LAN2:** https://192.168.2.12

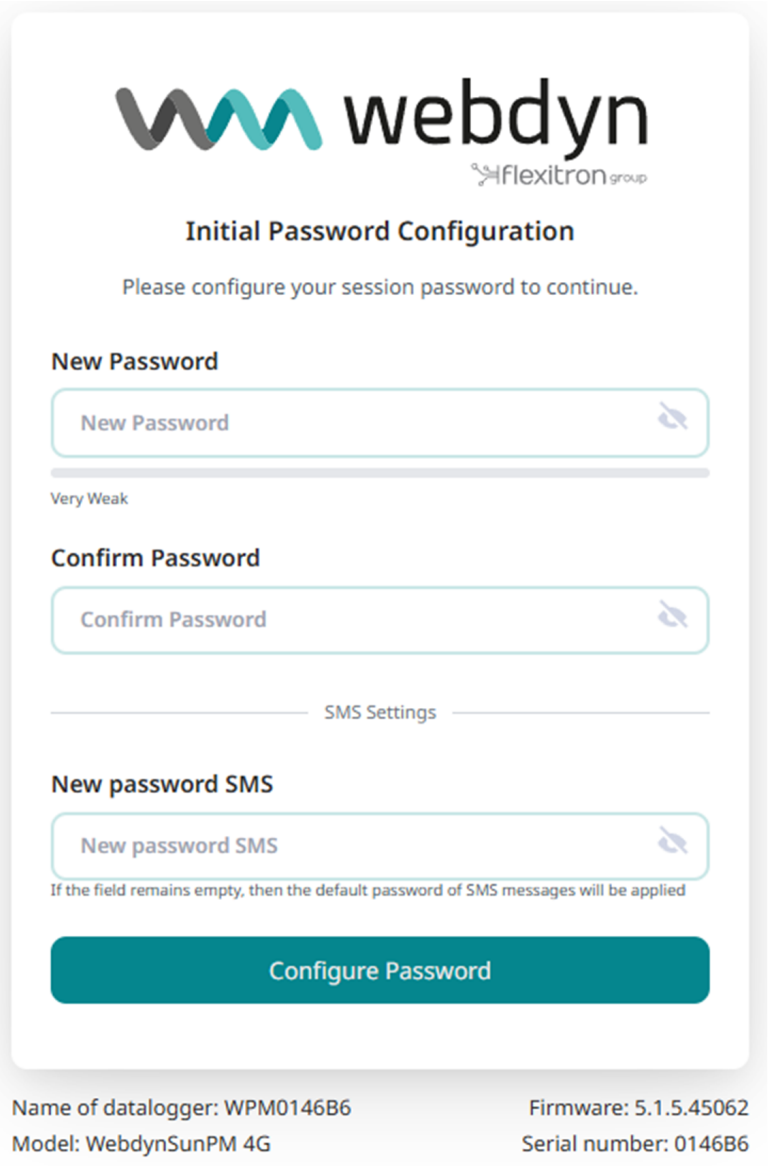
Warning

When connecting to the gateway via the HTTPS protocol, the browser may display a security warning. This message appears because the gateway uses a self-signed certificate. The first time you connect, you'll need to access your browser's advanced settings and manually approve the security exception to allow access to the interface.

Warning

The gateway remains accessible locally via unsecured connections at the following addresses: http://192.168.1.12 for LAN1 and http://192.168.2.12 for LAN2, but only if no gateway has been configured in the LAN settings.

If the connection is established, the login page appears



first connection

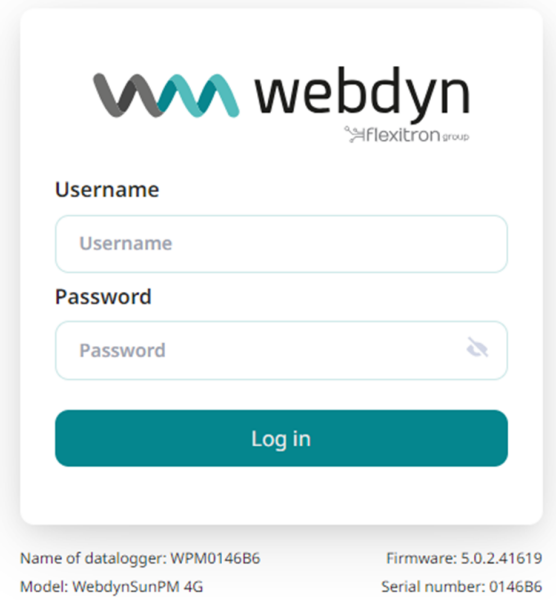
You are then asked to:

- Set the password for accessing the **(MANDATORY)** interface
- Set the SMS encryption password **(OPTIONAL)**

The strength of the access password is indicated and must have a minimum score of 4, calculated as follows:

- 8 characters +1
- 12 characters +1
- At least a number plus 1
- At least one +1 symbol
- Mixed case +1

A login window appears:



login

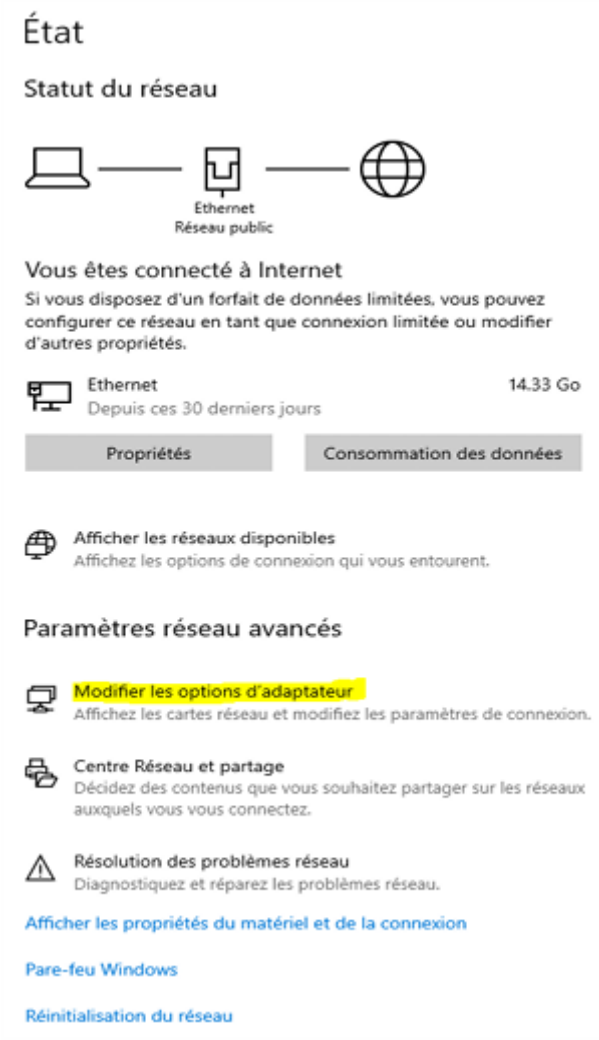
- **Username**: userhigh
- **Password**: Password set

> **Warning**
>
> If the page does not load, you will need to change your computer's IPv4 settings. Administrator privileges are required to make these changes

Changing the IP Address in Windows 10

To change your computer's IP address, right-click the network icon in the lower-right corner of your taskbar, then click **Network Parameters**

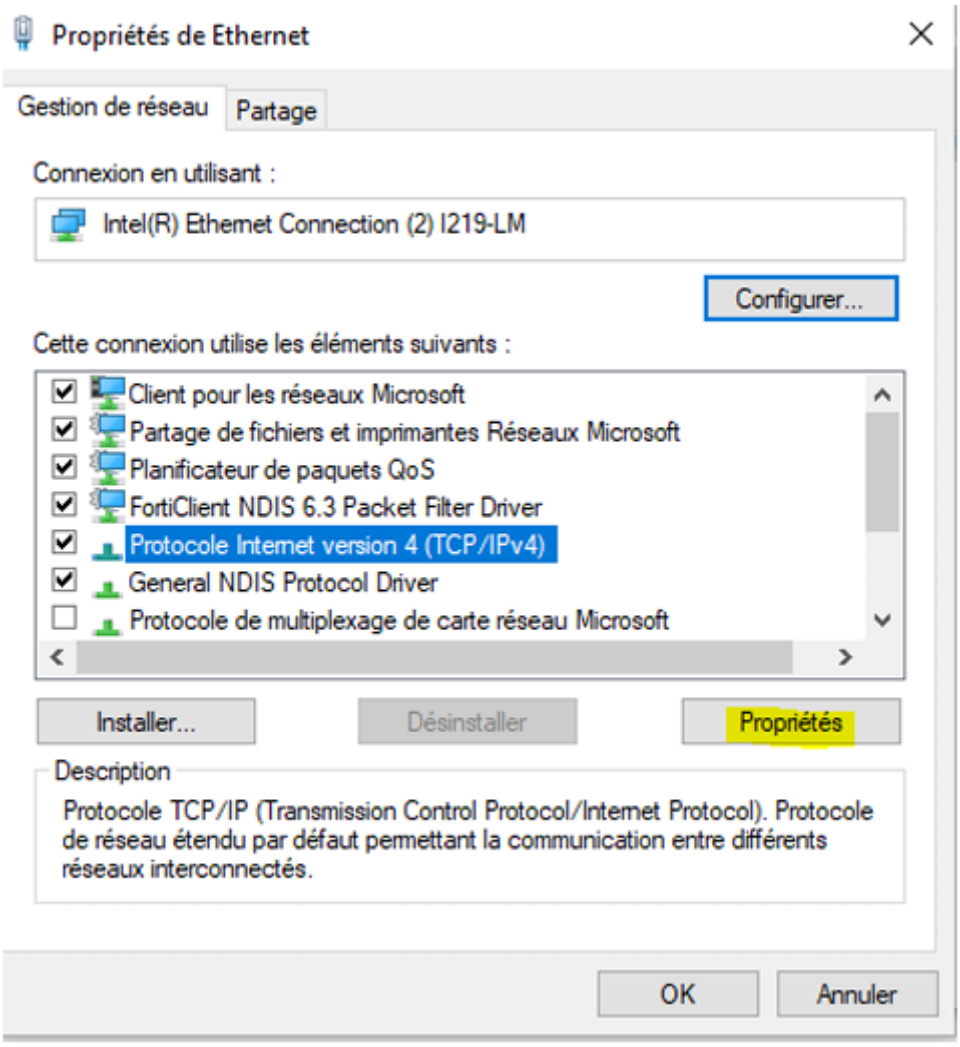
Click on **Change adapter settings** in the **Advanced network configuration** submenu:



win 10 network

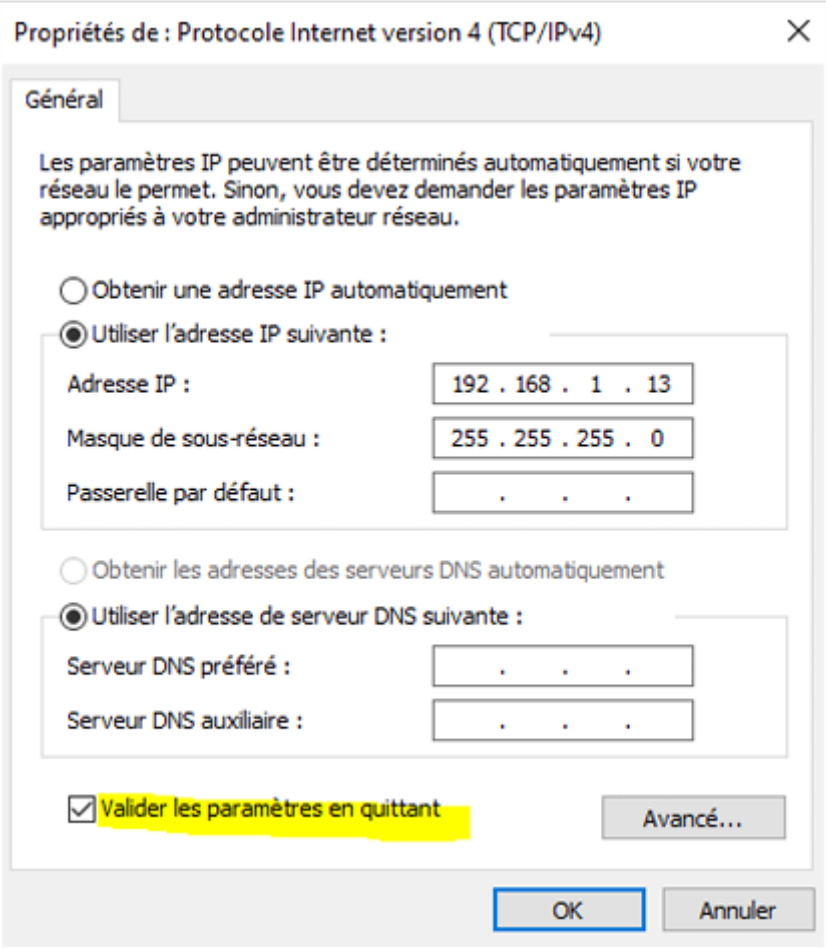
The network interfaces page will appear; right-click on the Ethernet interface, then select **Properties**.

The interface properties page appears; click **Internet Version 4 (TCP/IPv4)**, then **Properties**:



win 10 tcpip

The IP address configuration page appears:



win 10 tcpip 2

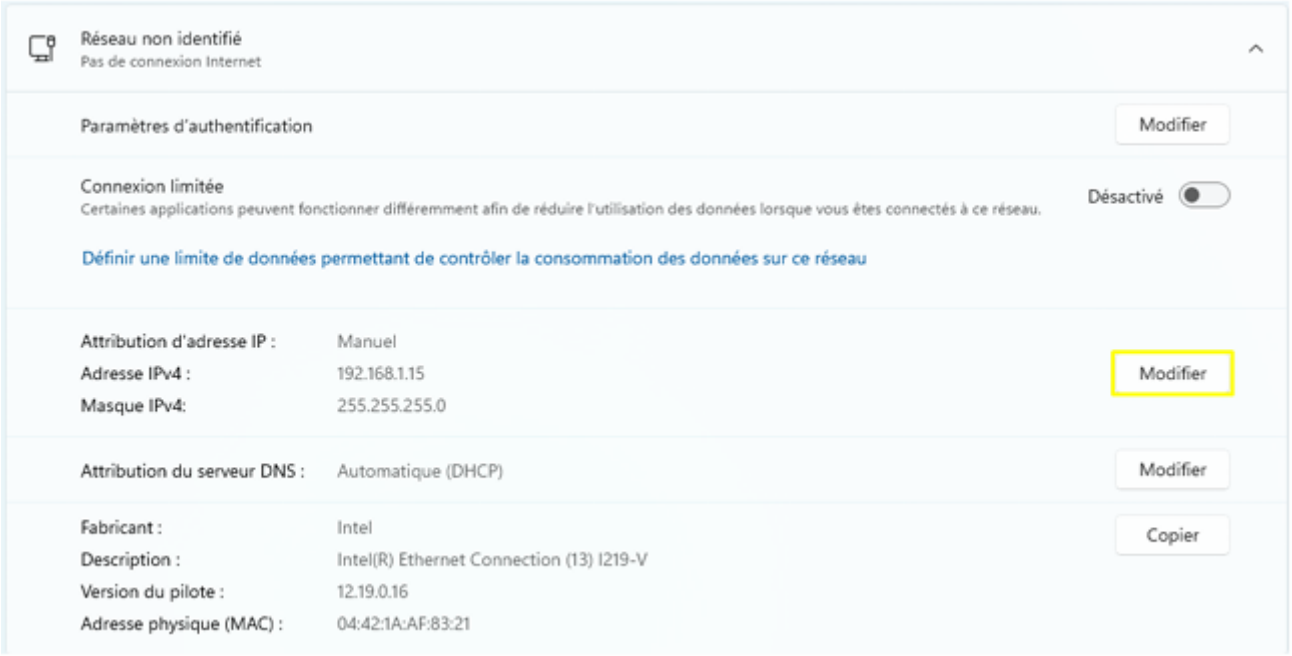
- Select the **Use IP Address** parameter
  - Enter the desired IP address (**192.168.1.x** for LAN1 or **192.168.2.x** for LAN2)
  - Subnet mask: **255.255.255.0**

Check the **Save on exit** box, then click **OK**

Changing the IP Address in Windows 11

To change your computer's IP address, right-click the network icon in the lower-right corner of your taskbar, then click **Network Parameters**

Click **Ethernet**, then click the **Change** button on the Ethernet interface:



win 11 network

The configuration page will appear. Enable IPv4, then enter the IP settings:

win 11 tcpip

- Enter the desired IP address (**192.168.1.x** for LAN1 or **192.168.2.x** for LAN2)
- Subnet mask: **255.255.255.0**

Click on **Save**

4.1.2 Quick Setup

A configurator is available on the home page to quickly configure the gateway and guide the installer.

The configurator consists of 6 steps:

- "**Network**": Allows you to configure the Ethernet and modem interface,
- "**Server**": Allows you to configure the connection to remote servers,
- "**Serial**": Allows you to configure the serial ports on the gateway,
- "**Detect or add device**": Allows you to add or detect devices connected to the gateway,
- "**Control**": Enables the services on the gateway,
- "**Change password**": Allows you to change the web interface password.

For the initial setup, the essential steps are network configuration and server configuration.

4.2 Configuring SFTP/FTP/WebDAV Servers

Using an FTP/SFTP/WebDAV-HTTPS server is the preferred solution for the remote configuration and operation of equipment.

> **Note**
>
> The use of FTP servers is not recommended for security reasons.

The gateway allows you to configure two remote servers.

If only one server is used, it can be configured as either Server 1 or Server 2, depending on the desired behavior.

- **Server 1**: primary server.
- **Server 2**: backup server.

The main server allows you to:

- Creating and modifying the configuration
- Order Shipping
- Alarm Reception
- Submitting and updating files

The backup server serves solely as a replica of the files stored on the main server.

No configuration changes or commands can be performed through this server.

4.2.1 Authentication and Security

Each server must be accessible using:

- a username
- a password.

> **Warning**
>
> SSH key authentication is not available.

The server can be hosted on any type of server (Windows, Linux, etc.). The important thing is that its address be accessible to the gateway using the configured network interface (Ethernet or modem).

Configuration serveur SFTP

Parameter	Specifications
Key Encryption Algorithm	`ssh-rsa`
Key exchange (KEX)	`diffie-hellman-group-exchange-sha256`
Encryption (ciphers)	`aes128-ctr` or `aes256-ctr`.
MAC (integrity)	`hmac-sha2-256` or `hmac-sha2-512`.
Compression	`none`

Configuration serveur WebDAV (HTTPS)

Parameter	Specifications
Protocol	WebDAV over HTTPS (https://)
Port	443 by default; can be configured if the server uses a different port
Authentication	Username and password
TLS Certificate	Valid server certificate for the configured address
Required Methods	HEAD, GET, PUT, DELETE
HTTP Metadata	The server must provide the Last-Modified and Content-Length headers for the files

4.2.2 Tree Structure

It should be noted that the gateway does not create the server directory structure. Therefore, you must manually create all directories. By default, at a minimum, the following directories must be present on the server:

File type	Default Folder
Configuration	/CONFIG
Alarms	/ALARM
Newspapers	/LOG
FW Update File	/BIN
Certificates	/CERT
Data Files	/DATA
Command Files	/CMD
Definition File	/DEF
Scripts	/SCRIPT

> **Note**
>
> These directories must have read, write, delete, and create permissions for the configured user

4.3 Web Server

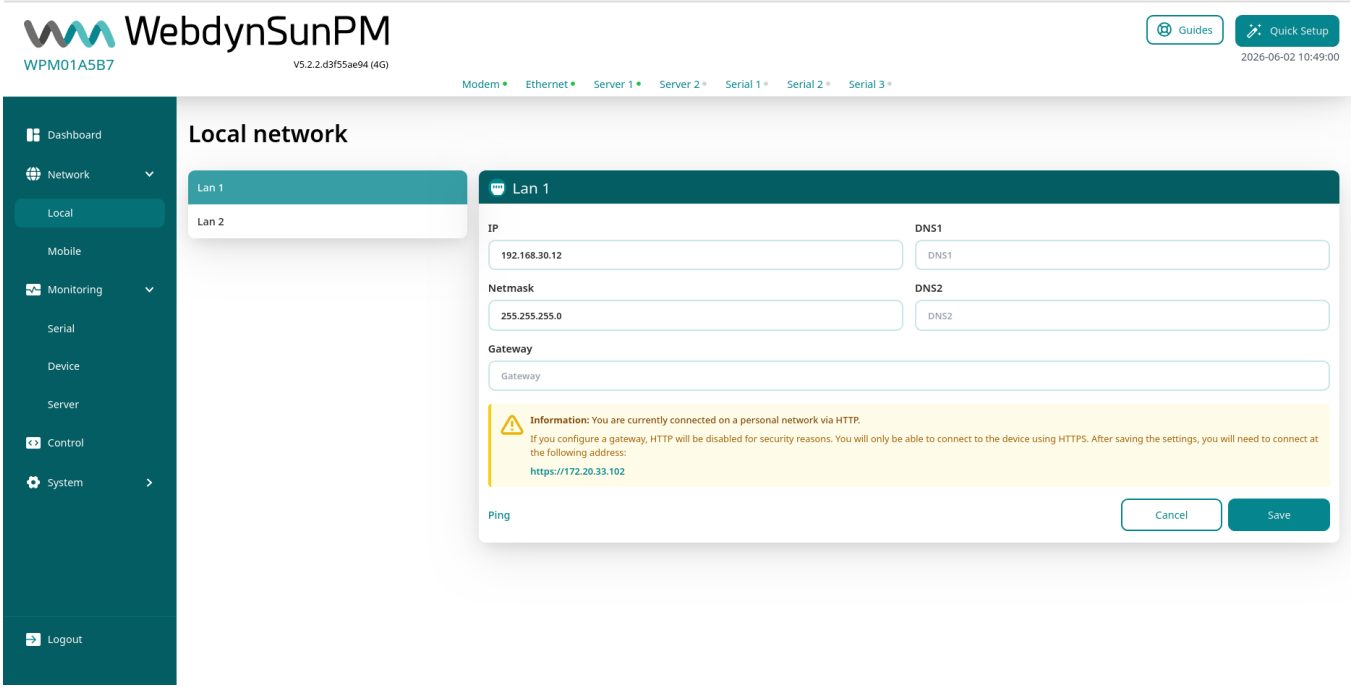
4.3.1 Configuring Network Interfaces

The first step is to configure the interfaces—either the mobile network or the LAN1/2 Ethernet interfaces, as desired.

> **Note**
>
> The mobile network is reserved exclusively for remote connections, while the LAN1/2 Ethernet interfaces can also be used to collect data from Modbus TCP devices

Ethernet (local)

From the **Network>Local** tab, the interface allows you to configure the two Ethernet interfaces (LAN1 and LAN2) available on the gateway. These Ethernet interfaces allow the gateway to be part of two different Ethernet networks.



network page

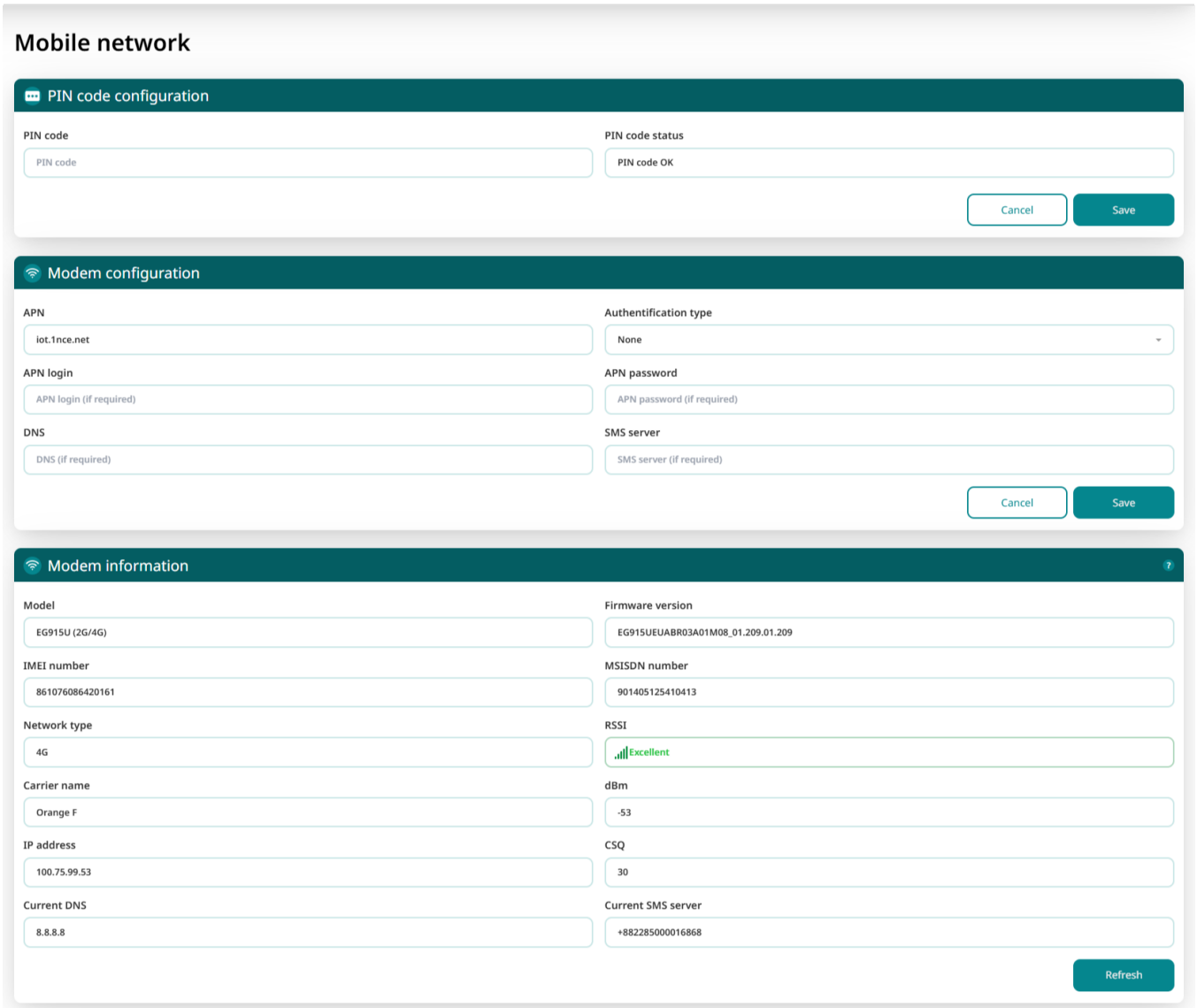
Web Interface	<UID>_daq.csv	Description
IP	ip	IP Address
Netmask	mask	Subnet mask
Gateway	gateway	Gateway Address
DNS 1	dns1	DNS Server 1
DNS 2	dns2	DNS Server 2

A button labeled **Ping** allows you to test connectivity to the gateway.

> **Warning**
>
> **Depuis FW 5.1.5**
>
> If a gateway is configured, the web server can only be reached via HTTPS on that interface

Mobile (modem)

From the **Network>Mobile** tab, the interface allows you to configure the modem to establish a remote connection. You must first insert the SIM card.



mobile page

Web	<UID>_daq.csv	Description
PIN code	pine	PIN
APN	apn	Name of the APN
Authentication Type	authentication	Type of authentication with the carrier (optional, depending on the carrier):
APN Login	login	Your mobile carrier's username (optional, depending on the carrier)
APN password	password	Your mobile carrier's password (optional, depending on the carrier)
SMS server	server	SMS messaging center. (optional)
DNS	DNS	DNS server (optional)

The different PIN statuses are as follows:

- **Pin Code OK**: The modem can access the SIM card
- **Pin code Required**: The SIM card is waiting for a code
- **Unkown**: Various modem errors
- **PUK code required**: The SIM card is locked
- **SIM card not inserted**: No SIM card inserted

> **Note**
>
> If no APN is specified, no IP connection is possible. Only SMS

Using SMS commands, you can configure the product remotely, provided that a SIM card is inserted and there is no PIN code. A specific application note is available.

Modem information is also available:

- **Firmware version**: Modem software version
- **IMEI number**: Unique modem identification number
- **MSISDN number**: Identification number of the inserted SIM card
- **Network Type**: Type of network to which the modem is connected (2G/3G/4G).
- **RSSI**: Indication of the modem's receive power level
- **Carrier name**: Operator Name
- **dBm**:Signal level
- **IP Address**: IP address
- **CSQ**: Received signal level between 0 and 31
- **Current DNS**: The DNS address currently used by the modem connection.
- **Current SMS server**: SMS messaging center currently in use.

The **Refresh** button forces the modem information to be updated, which is useful when the SIM card has just been inserted.

4.3.2 Configuration of the Main and Secondary Servers

Two servers can be configured on the gateway. The first, named **Server 1**, is the main server that enables full remote administration of the gateway; the second, named **Server 2**, is the secondary server, which acts as a mirror of the main server.

> **Info**
>
> The web server allows you to configure the `<UID>_config.ini` configuration file

The protocols available for each server vary:

Server	Protocol available
Server 1 (main server)	SFTP, FTP, WebDAV
Server 2 (secondary)	SFTP, FTP, WebDAV, MQTTs, MQTT, Azure MQTT, AWS MQTT

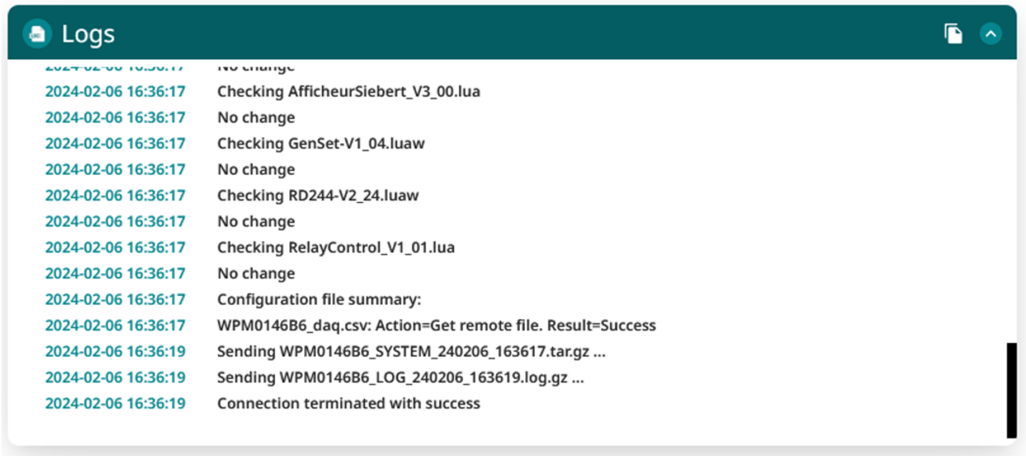
> **Warning**

> The FTP and MQTT protocols are not considered secure; it is recommended to use SFTP, MQTTs, or WebDAV.

You can choose the connection interface for each server:

	Server 1	Server 2
Modem	✓	✓
Ethernet	✓	✓
SD card	✓	✗

To verify that the server configuration is correct, we recommend clicking the **Connect** button. A window displaying the connection log will appear, showing all the files exchanged between the gateway and the remote server. The last line lets you quickly see whether the connection was successful or failed.



log server connection

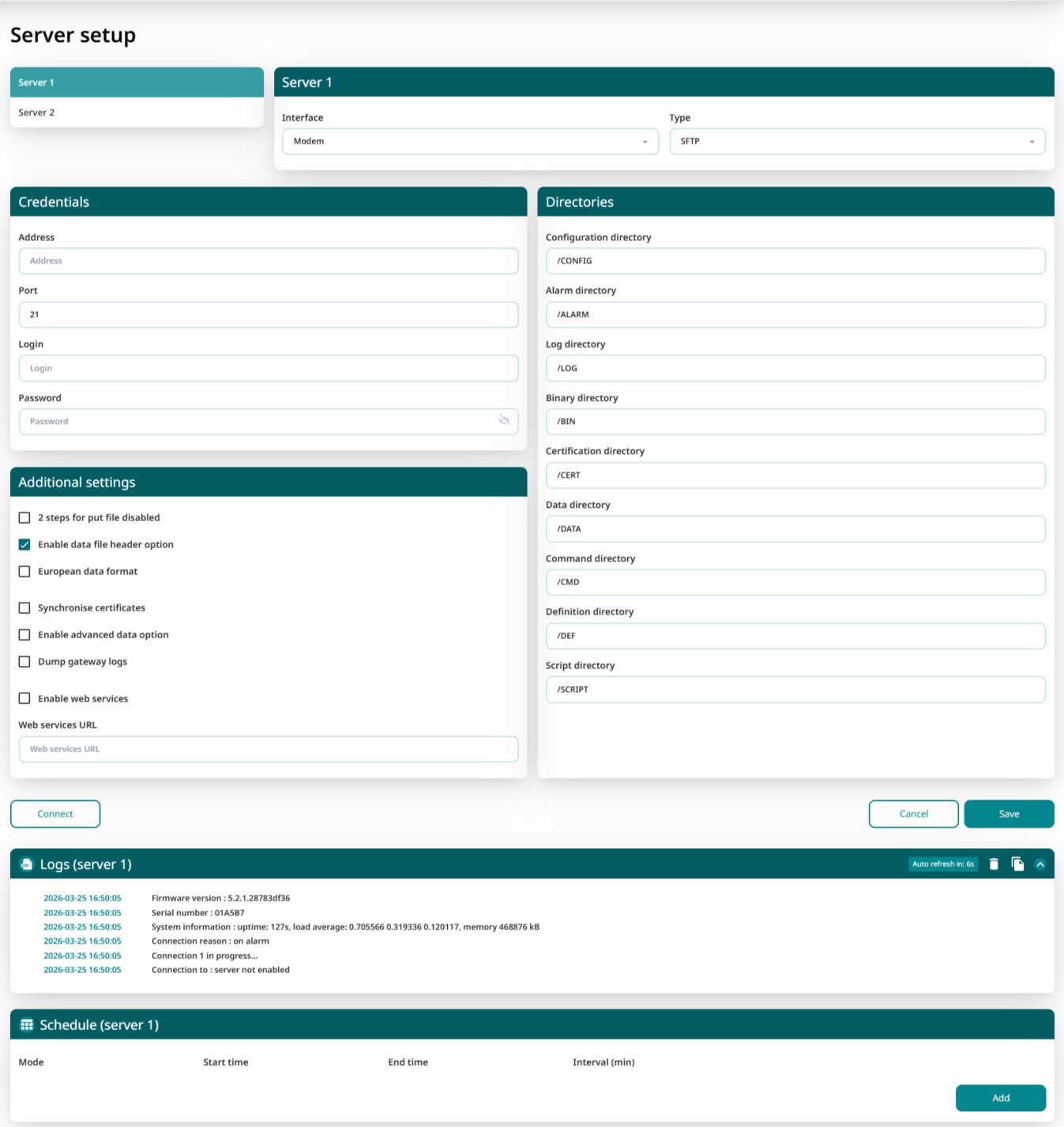
Interface and Protocol

With `SERVERSLOT = { SERVER, SERVER2 }`

Web	<uid>_config.ini	Description
Interface	<SERVERSLOT>_Interface	Selecting the Network Interface
Type	<SERVERSLOT>_Type	Protocols

SFTP/FTP/WebDAV Server

These three types of servers are similar:



server page

You need to perform configuration on the following sections:

- Interface and Protocol
- Access (Credentials)
- Directories
- Additional Settings

Access (Credentials)

With `SERVERSLOT = { SERVER, SERVER2 }`

With `SERVER = { FTP, FTP2, HTTP, HTTP2 }`

Web	<uid>_config.ini	Description
Address	<SERVERSLOT>_Address	Server IP address or URL
Port	<SERVER>_Port	Server port
Login	<SERVER>_Login	Username
Password	<SERVER>_Password	Password

Directories

With `SERVER = { FTP, FTP2, HTTP, HTTP2 }`

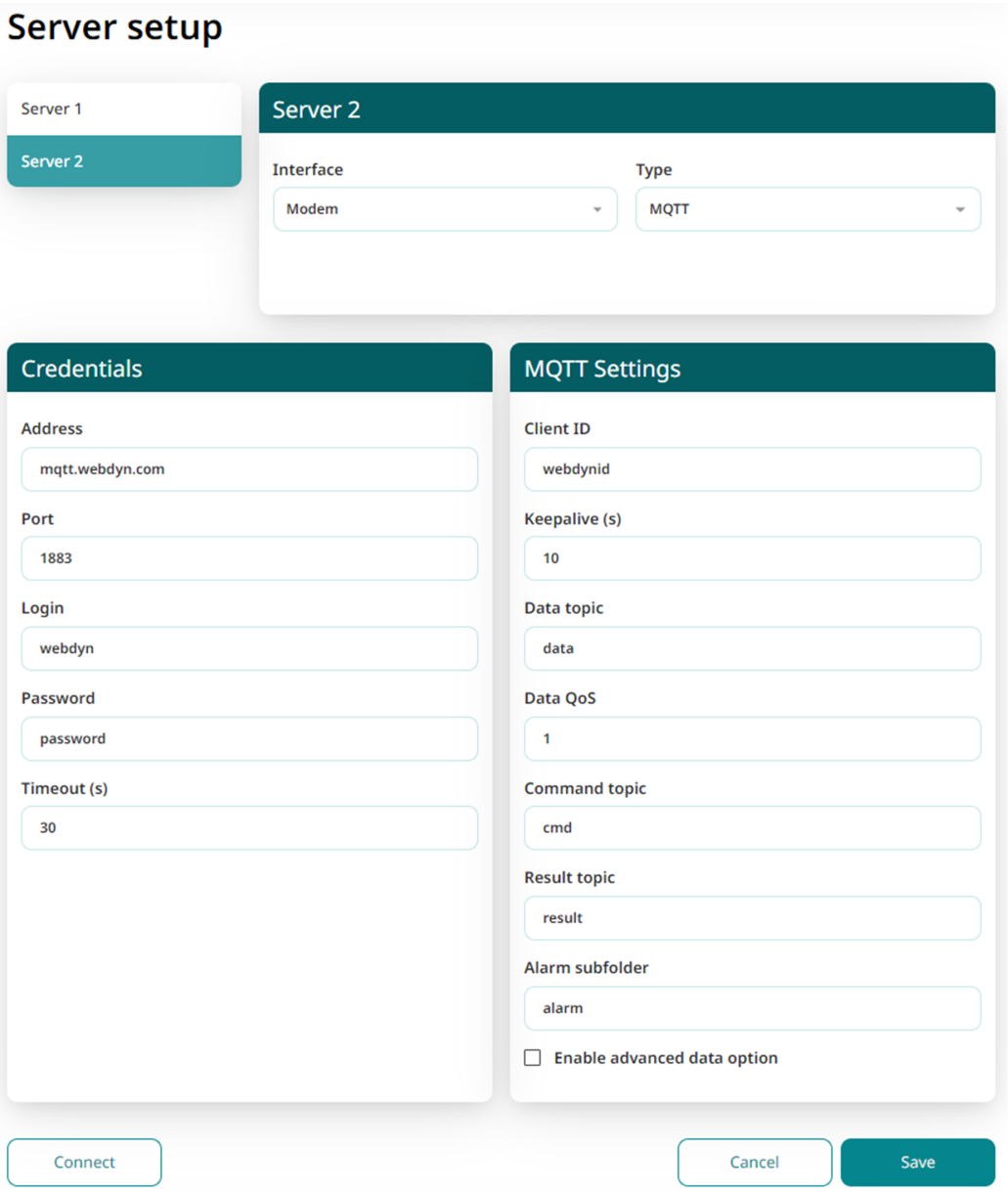
Web	<uid>_config.ini	Description
Configuration directory	<SERVER>_DirConfig	Directory of Configuration Files
Alarm Directory	<SERVER>_DirAlarm	Directory of Alarm Files
Log directory	<SERVER>_DirLog	Log Directory
Binary directory	<SERVER>_DirBin	Directory of Update Files
Certificate Directory	<SERVER>_DirCert	Directory of certificate files
Data directory	<SERVER>_DirData	Data Files Directory
Command directory	<SERVER>_DirCmd	Directory of Command Files
Definition directory	<SERVER>_DirDef	Directory of Definition Files
Script directory	<SERVER>_DirScript	Script File Directory

Additional Settings

With `SERVER = { FTP, FTP2, HTTP, HTTP2 }`

Web	<uid>_config.ini	Description
2 steps to disable the put file	<SERVER>_TwoStepsSendingDisabled	2-step file transfer
Enable the data file header option	<SERVER>_HeaderOption	Optional headers in data files
European date format	<SERVER>_EuroDateFormat	Date Format
Synchronize certificates	<SERVER>_SynchronizeCertificates	Certificate Synchronization
Enable advanced data option	<SERVER>_EnableAdvancedData	Added the number of complete reads
Dump gateway logs	<SERVER>_UploadLog	Uploading system log files
Enable Web Services	<SERVER>_WebServicesEnable	Enabling Web Services
Web Services URL	<SERVER> _WebServicesUrl	URL called by web services

MQTT/MQTTs/MQTT on Azure/MQTT on AWS



mqtt server page

Access (Credentials)

Web	<uid>_config.ini	Description	MQTT	MQTTs	Azure MQTT	MQTT on AWS
Address	SERVER2_Address	IP address or server name	✓	✓	✓	✓
Port	MQTT2_Port	MQTT server port (default 1883)	✓	✓	✓	✓
Login	MQTT2_Login	Username	✓	✓	✗	✗
Password	MQTT2_Password	Password	✓	✓	✗	✗
Timeout (s)	MQTT2_Timeout	Maximum waiting period	✗	✓	✓	✓
Insecure	MQTT2_Insecure	Verification of the hostname specified in the certificates	✗	✓	✗	✗

Note

AWS MQTT and Azure MQTT require connection via a key and authentication via a certificate only

MQTT/MQTTs Settings

Web	<uid>_config.ini	Description	MQTT	MQTTs	MQTT on AWS
Client ID	MQTT2_ClientId	Gateway ID	✓	✓	✓
Keepalive(s)	MQTT2_KeepAlive	inactivity timeout	✓	✓	✓
Data Topic	MQTT2_Topic	Topic name for data submitted by the gateway.	✓	✓	✓
Data QoS	MQTT2_QoS	Guaranteed service level for message delivery (Quality of Service)	✓	✓	✓
Command topic	MQTT2_ControlTopic	Topic name for commands to be retrieved by the gateway	✓	✓	✓
Result topic	MQTT2_ResultTopic	Topic Name for Command Results	✓	✓	✓
Alarm Topic	MQTT2_AlarmTopic	Name of the alarm topic you want to publish	✓	✓	✓
Enable advanced data options	MQTT2_EnableAdvancedData	Publication of the number of complete readings	✓	✓	✓

Note

If the command topic is specified, the connection to the MQTT server will be persistent so that commands sent can be received.

The Azure MQTT settings are specific:

MQTTs Azure:

Web	<uid>_config.ini	Description
Cloud IoT Hub	MQTT2_CloudProjectId	Unique project identifier
Cloud device	MQTT2_CloudDevice	Unique Device Identifier
Keepalive(s)	MQTT2_KeepAlive	inactivity timeout
Data QoS	MQTT2_QoS	Quality of Service
Enable advanced data options	MQTT2_EnableAdvancedData	Publication of the number of complete reads
Enable method invocation	MQTT2_EnableInvokeMethod	Enable method calls
Publish Alarms	MQTT2_EnableAlarmPost	Enable alarm posting to the dedicated topic

Certificates

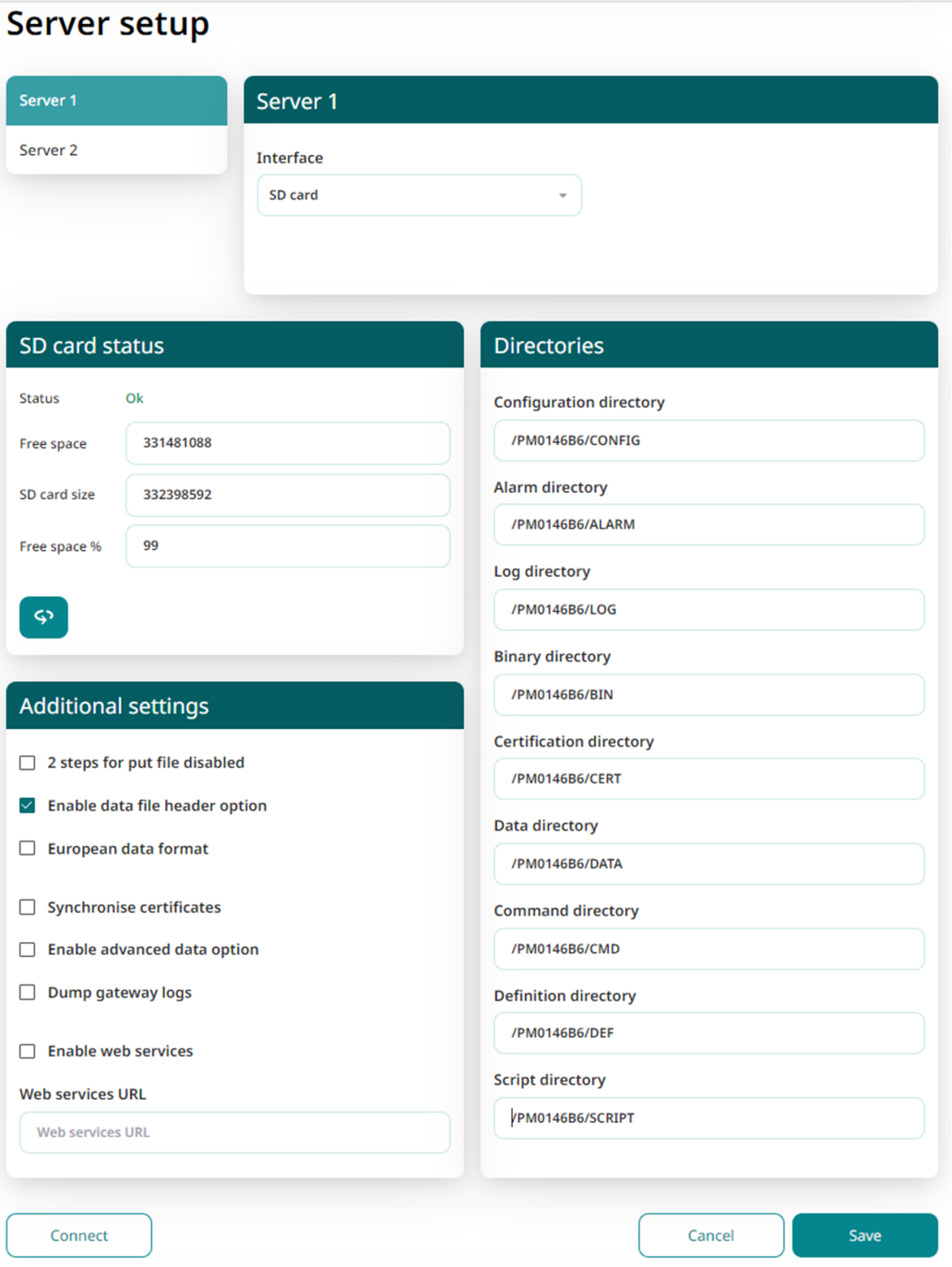
Web	<uid>_config.ini	Description
SunPM Private Key	MQTT2_KeyFile	File containing the gateway-specific private key
CA certificate	MQTT2_CaCertFile	MQTTS server certificate entered
SunPM Certificate	MQTT2_CertFile	Certificate specific to the gateway used for the connection

> **Note**
>
> Certificates must be retrieved from the MQTT servers and imported into the gateway either via the web server, as shown here, or directly into the certificate folder (by default `/DEF`) on the main server.

SD card configuration

The SD card is available only on Server 1. The SD card is not a remote server and does not require any special access configuration.

The information related to the SD card is then displayed:



sd card page

> **Note**
>
> If the directories do not exist on the SD card, they will be created automatically by the gateway.

The information box on the SD card contains the following information:

- **Status**: SD card status; possible statuses are:
  - **Unknown**: The status is unknown because there has been no access to it so far
  - **Ok**: The last attempt to write to the SD card was successful
  - **Failed** to mount SD Card: The SD card was not detected. Check that the card is properly inserted and that it is formatted correctly (FAT32 or exFAT).

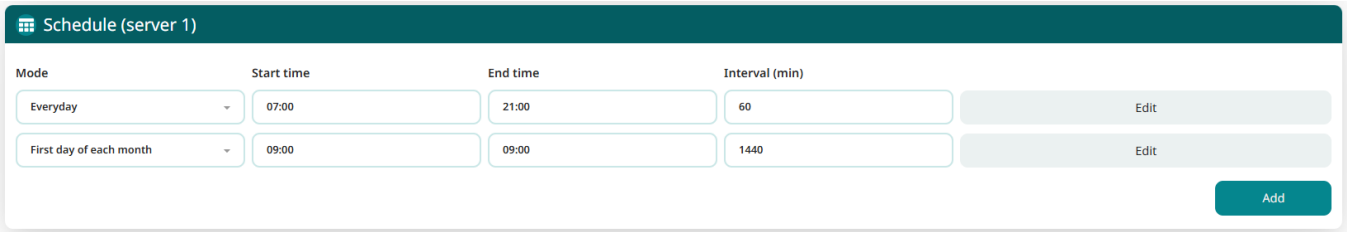
    - **SD Card read error**: The SD card has been detected, but errors occurred while reading the information on it. Check to make sure the card is properly formatted (FAT32 or exFAT).
- **Free space**: Available memory space on the SD card (in bytes)
- **SD card size**: SD card capacity (in bytes)
- **Free space (%)**: Percentage of SD card usage. Allows you to quickly and easily monitor how full the SD card is.

> **Note**
>
> You can load a new configuration onto the gateway by copying the configuration files directly to the SD card before inserting it into the gateway.

Planning Server Connections

On each server, you can schedule how often files are uploaded from the gateway.

> **Note**
>
> The web server allows you to configure the `<UID>_var.ini` scheduling file

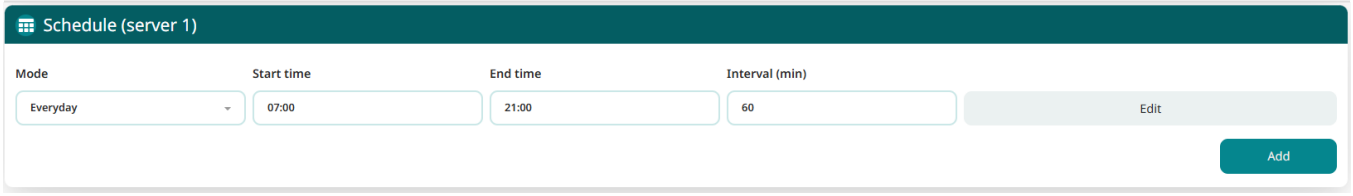


scheduler page

Web	<uid>_var.ini	Description
Fashion	fashion	Scheduling mode (daily, weekly, monthly)
Start time	StartTime	Start Time for Transmission
End Time	-	Kickoff Time
Interval	interval	Time between two connections

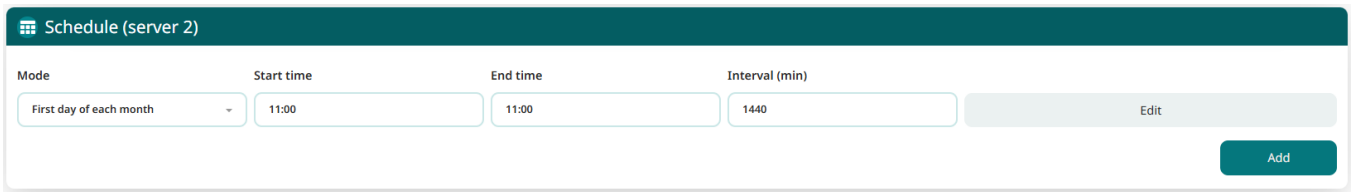
> **Note**
>
> The value **End Time** automatically adjusts after the schedule is confirmed based on the number of possible intervals; it will indicate the time of the last occurrence of the day.

The example below shows a daily file-sending schedule that begins at 7 a.m. and ends at 9 p.m., with files sent every hour.



scheduler page 1

The example below shows a monthly schedule for sending files.



scheduler page 1

It is also possible to have multiple schedules per server.

Tip

For a photovoltaic plant, file uploads can be disabled overnight

Note

The first connection takes place at least 5 minutes after the product's startup.

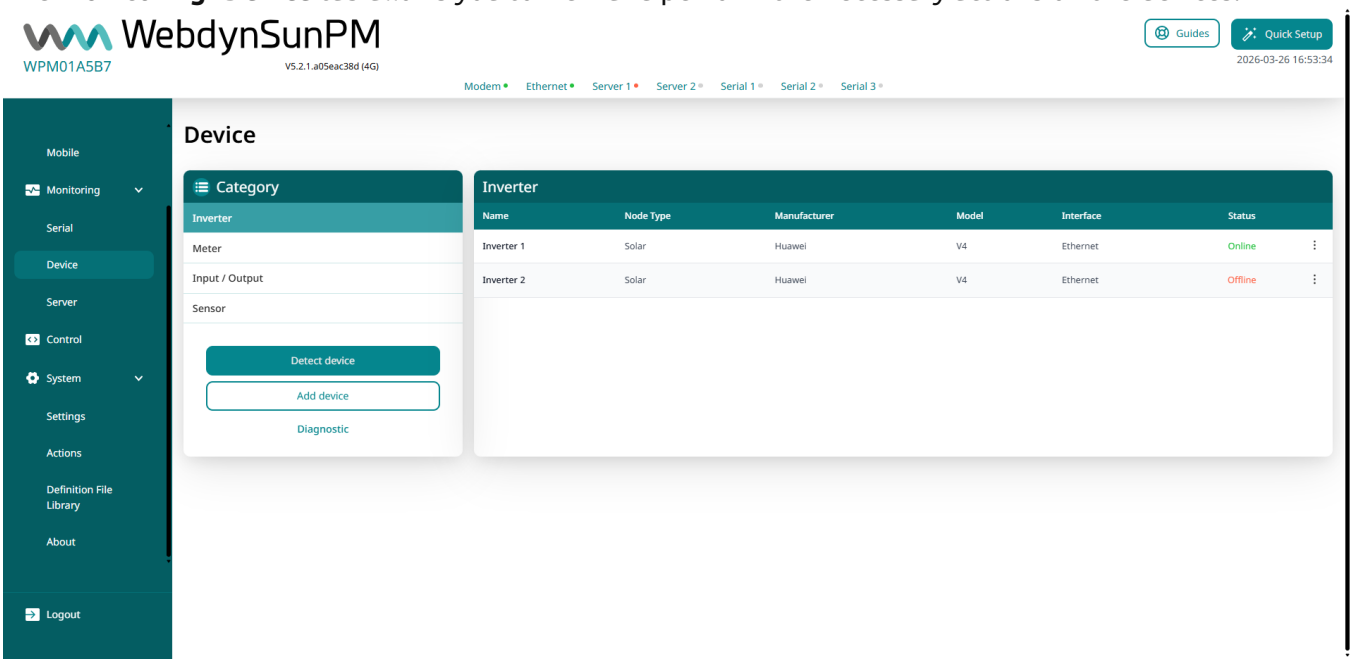
4.3.3 Device Configuration

From the web server, you can either automatically add devices through auto-detection or manually register them.

Note

It is also possible to add devices from the main server or via text message. These procedures are described in their respective sections.

The **Monitoring>Device** tab allows you to view and perform the necessary actions on the devices:



> **Note**
>
> The devices are filtered by category.

From this tab, you can access information about the devices that have already been configured, presented in a table format:

- **Name**: Device name. Field `name` in the configuration file `<UID>_daq.csv`
- **Node Type**: The application node. Field `node` in the configuration file `<UID<_daq.csv`
- **Manufacturer**: Manufacturer. Field `Manufacturer` in the device definition file
- **Model**: Model/Variant. Field `model` in the device definition file
- **Interface**: communication interface. Field `interface` in the configuration file `<UID>_daq.csv`
- **Status**: Communication status with the device

Automatic Detection

> **Note**
>
> Not all devices are compatible with detection

The devices compatible with automatic detection are listed below:

Protocole Modbus

- SunSpec (RTU/TCP)

Proprietary Protocol

- SMA-net
- DIEHL
- Solarmax
- PowerOne
- Delta
- Kaco

- Fronius
- Siemens Refusol
- ICT

The **Detect Device** button allows you to perform configuration of automatic detection using the following settings:

- **Interface**: Affected interface and protocol
- **Number of devices**: Number of devices to detect
- **Timeout (ms)**: Timeout for device return

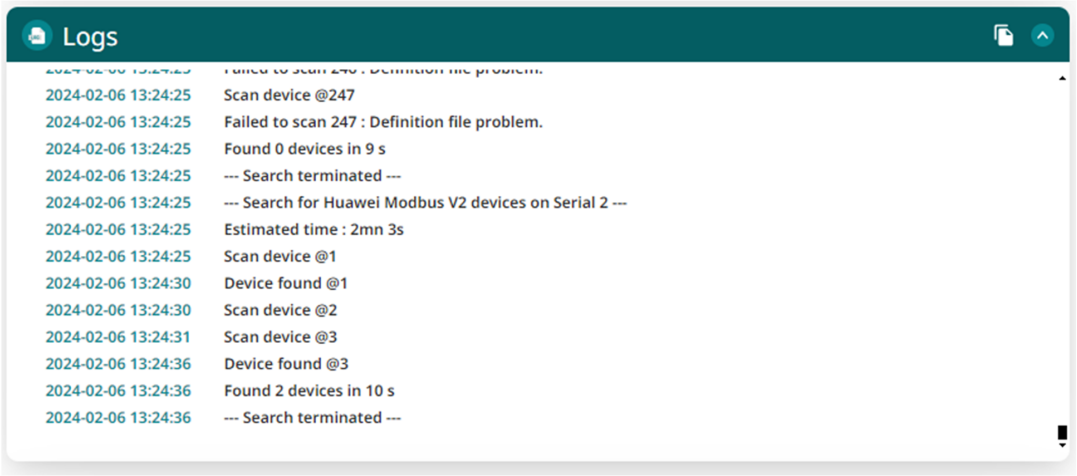
> **Note**
>
> It is recommended that you do not set the timeout value too high, so as not to slow down detection.

SunSpec Detection

> **Note**
>
> For SunSpec detection to work, the definition file library must be present on the gateway

> **Note**
>
> Make sure the serial or Ethernet configuration is correct

When you click the **Start detection** button, a progress window appears:



device page 2

The progress window displays the start date and time of the scan, as well as an estimated duration for the scan, in the first few lines.

The messages displayed allow you to track progress. The example given above relates to SunSpec detection.

As detection proceeds, the progress will be displayed on the web page with the following information:

NON SUNSPEC device found: A Modbus device (but not a SunSpec device) was detected at the specified address

```
2023-02-03 9:56:10:NON SUNSPEC device found at @1 192.93.121.23:502
```

Non-SunSpec device detected at IP address:port 192.93.121.23:502 on Modbus address 1

SunSpec device found: A Modbus device that meets the SunSpec specifications has been detected at the specified address

```
2023-02-03 09:56:21:SunSpec device found at @126 192.93.121.23:502
```

SunSpec device detected at IP address:port 192.93.121.23:502 on Modbus address 126

Found table: A SunSpec table has been detected. The information line then displays the table identifier, its size, and the affected device

```
2023-02-03 09:56:21:Found table 1:66 at 40004@126 192.93.121.23:502
```

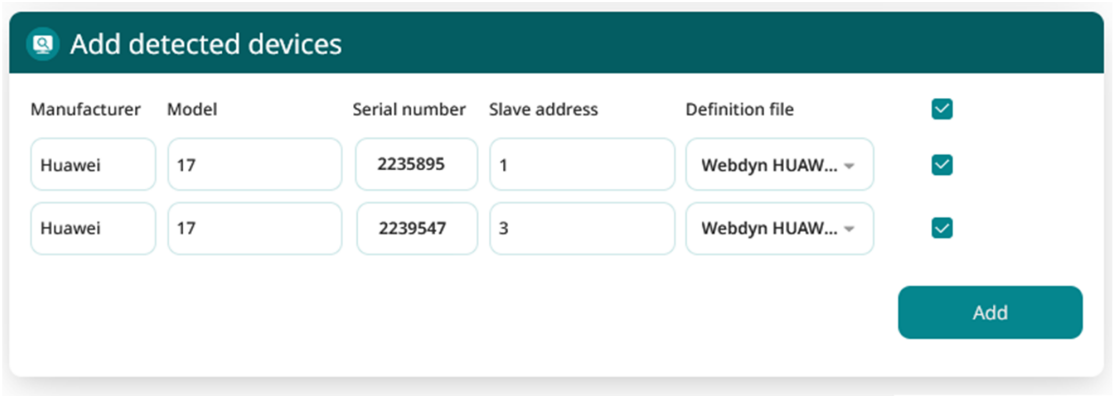
Table 1, consisting of 66 registers, was detected on the Modbus TCP device at IP address:port 192.93.121.23:502, in register 40004 and Modbus address 126.

End of SunSpec detection: SunSpec detection complete; number of devices found.

```
2023-02-03 09:56:53:End of SunSpec detection on ethernet1. 1 devices found.
```

1 SunSpec device detected on LAN interface 1.

At the end of the discovery process, the final page of the discovery process allows you to view all the devices that were discovered and, if desired, add them to the configuration.



device page 3

This screen displays the detection results with the following information:

- Manufacturer
- Model
- Serial Number
- Slave address
- Definition File

The selected devices will be added to the configuration. If a device has already been configured, it will be deselected by default.

> **Note**
>
> If the device already existed in the configuration and the user forces a new import, the previous device is not overwritten. A new device is created in addition to the preexisting one if the name is different.

The variables for this new device will be generated based on the detected SunSpec tables. Thus, for each detected SunSpec table, the following variables will be created:

- **<idTable>_tableId**: Numeric table identifier, as a 16-bit integer
- **<idTable>_tableSize**: table size in number of registers, as a 16-bit integer
- **<idTable>_<variableName>**: variable name consisting of the table identifier followed by the variable name
- **<idTable>*<repeatBlock>*<variableName>**: For variables that originate from a repeating block, the variable name consists of the table identifier, the repetition number, and the variable name.

By default, the generated variables are of type **Parameter** (action code **1**), except for:

- Table 101: Inverter (Single-Phase)
- Table 102: Inverter (Split Phase)
- Table 103: Inverter (Three-Phase)
- Table 111: Inverter (Single-Phase) - Float Value
- Table 112: Inverter (Split Phase) - Float Value
- Table 113: Inverter (Three-Phase) - Float Value
- Table 123: Immediate Controls
- Table 160: Multiple MPPT Inverter Extension Model
- Table 401: String Combiner (current)

These tables will be created with the type **snapshot** (action code **4**).

When the device is created, tags are automatically assigned to SunSpec variables:

Proprietary protocol detection

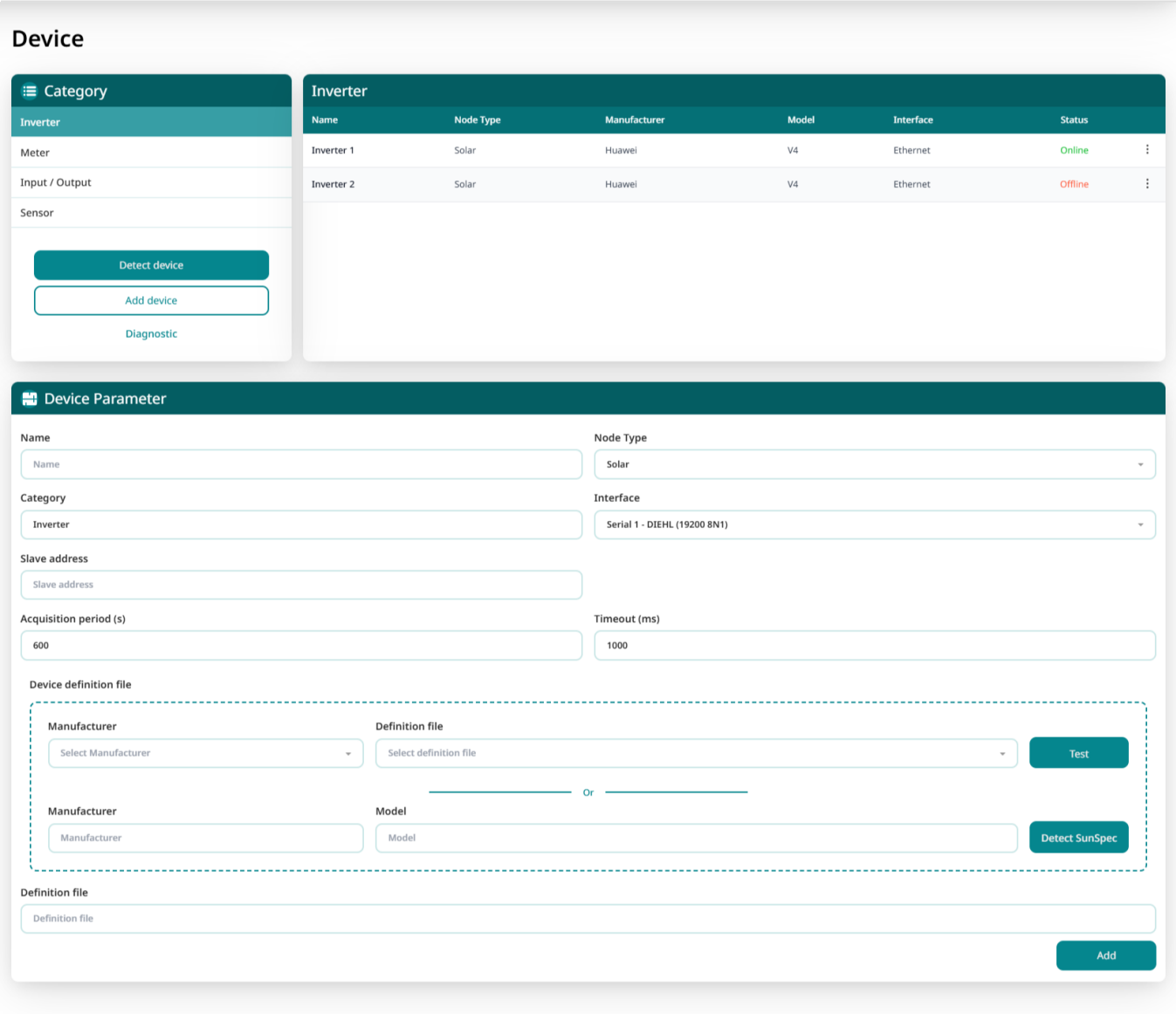
Automatic detection of proprietary protocols is described in the appendix for each protocol.

Manual Management of Devices

> **Note**
>
> The web server allows you to configure the `<UID>_daq.csv` configuration file

Add Devices

To add a device, select the category and then click **Add device**



device page 4

The device creation form appears with the settings corresponding to the configuration file `<UID>_daq.csv`:

- **Name**: Unique device name
- **Node Type**: Node type from a list
- **Interface**: The communication interface
- **IP address*: IP address (if using Ethernet)
- **Port**: Ethernet port
- **Slave address**: Modbus Address
- **acquisition period**: Acquisition period in the data file
- **timeout (ms)**: Timeout before the request fails

Node Type corresponds to the various types of nodes that functionally group devices.

The next step is to associate a definition file either manually or through SunSpec detection.

To set up the association manually, select the manufacturer and a definition file. A **Test** button allows you to test communication with the selected device.

> **Note**
>
> The list of definition files is filtered based on the selected category.

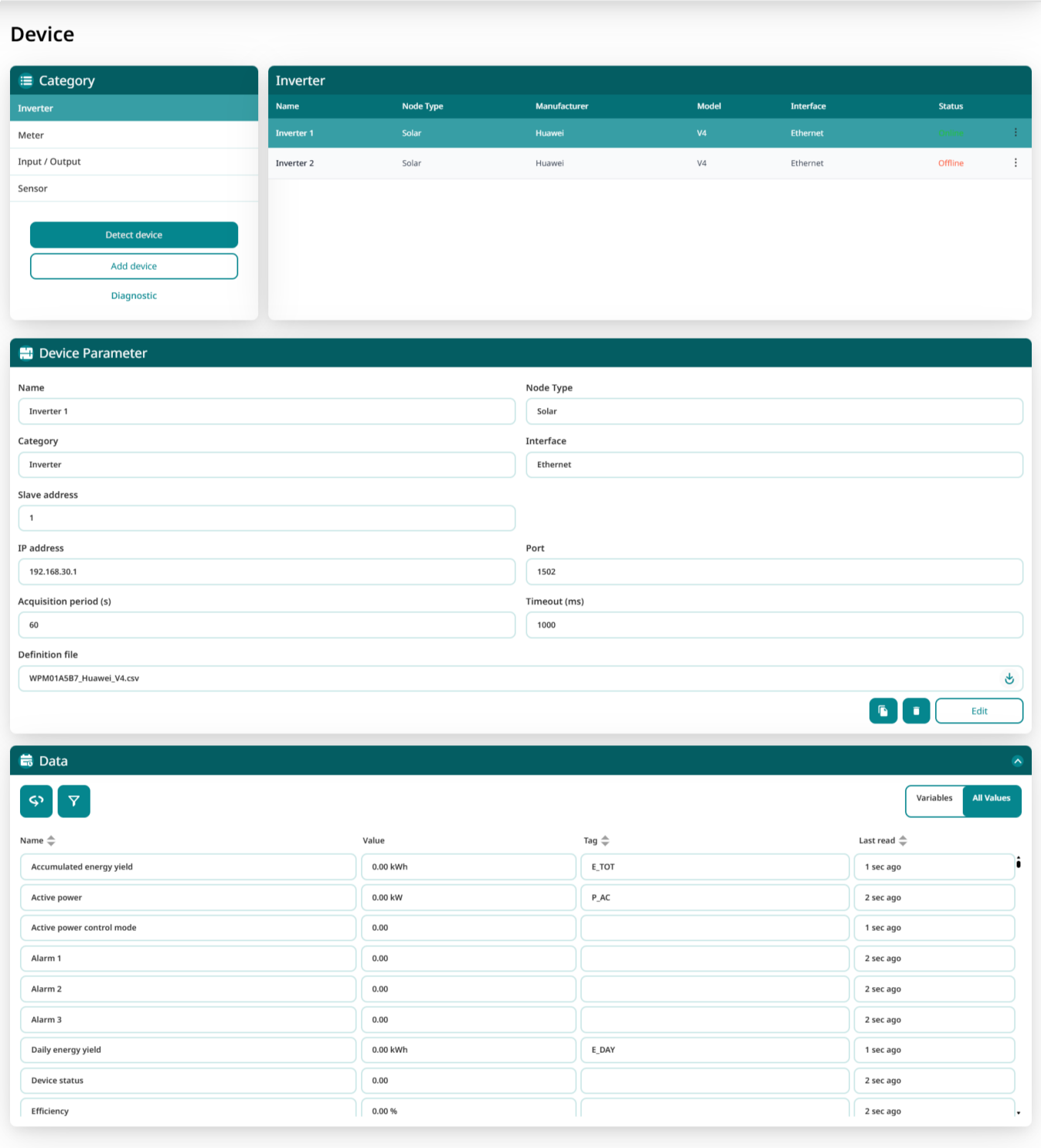
The addition of the device is confirmed by pressing **Save**

Edit/Duplicate/Delete Device

From **Monitoring>Device**, select the category of the device you want to edit.

In the **Device Parameters** section, there are three possible actions:

- **Duplicate**: The devices will be duplicated
- **Delete**: The device will be removed
- **Edit**: The device can be edited



device page 5

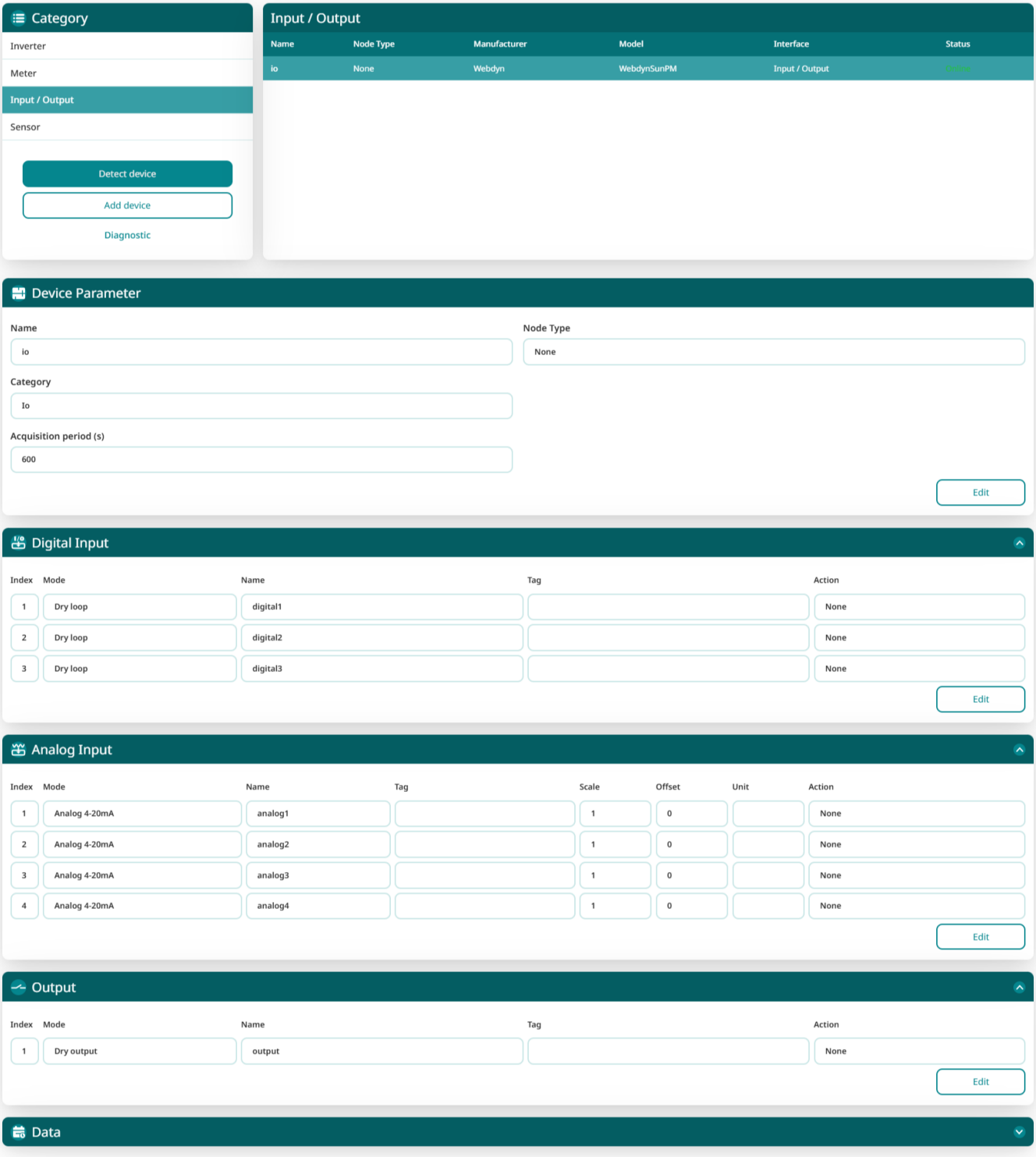
Edit Internal I/Os

The gateway's inputs and outputs appear as devices in the **Input / Output** category.

> **Note**
>
> Devices cannot be duplicated or deleted

You can edit this device to configure its inputs and outputs.

Device



device io page 1

Device Parameters

Name and **Node Type** cannot be edited.

Digital Input

For each of the three digital inputs, you can change the input type, name, tag, and action.

Analog Input

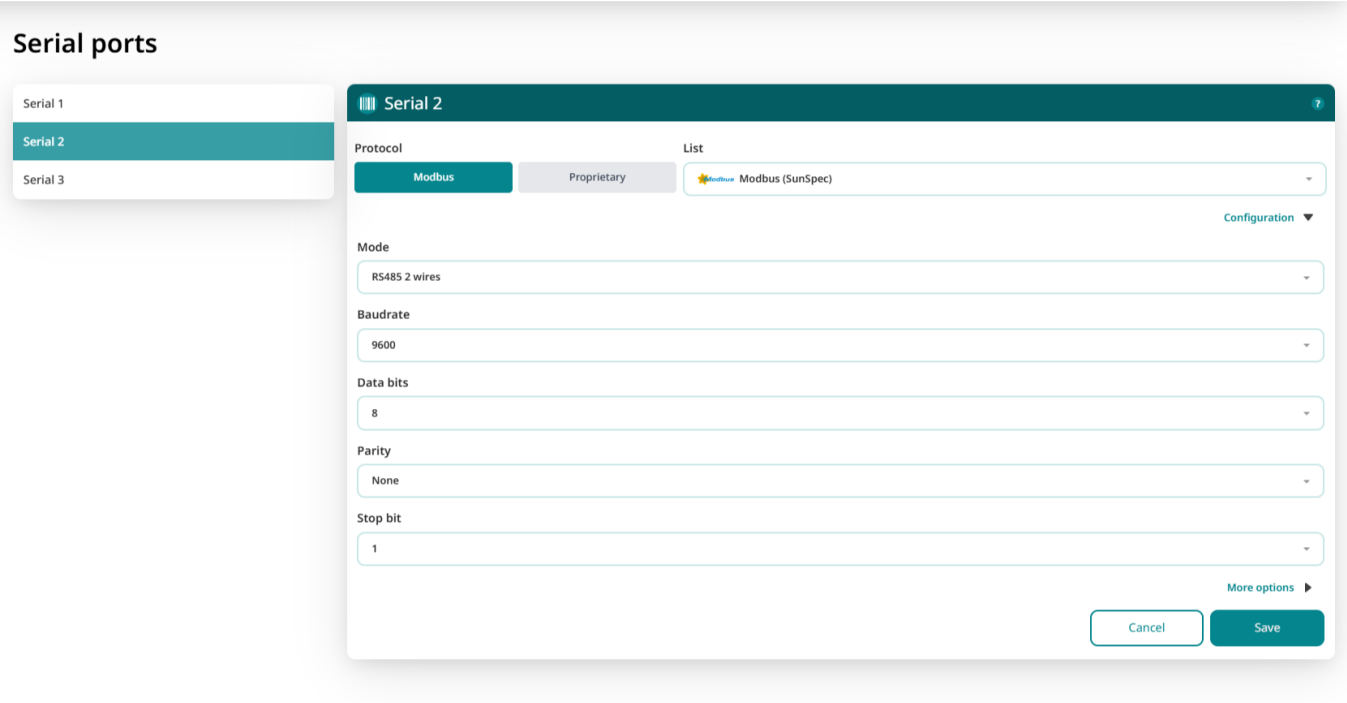
For each of the four analog inputs, you can change the input type (Mode), name, tag, scaling, unit, and action.

Output

For the internal relay, you can change the type (Mode), name, tag, and action.

4.3.4 Configuring Serial Connections

The **Monitoring>Serial** tab allows you to configure the three serial ports and serves as an interface to the `<UID>_daq.csv` configuration file.



serial page

Web Interface	Setting in the <uid>_daq.csv configuration file	Description
Set by device brand	protocol	Protocol type (Modbus, Modbus configured according to the manufacturer, proprietary protocol)
Fashion	wires	Serial Interface Mode
Baud rate	baud rate	Serial connection speed in baud
Data bits	data_bits	Number of data bits
Parity	parity	Serial Link Parity
Stop bits	stop_bits	Number of stop bits
Advanced Options	---	---
Inter-frame (ms)	interframe	Waiting time between two frames
Forwarded TCP port	forwarded_port	communication tunnel between a Modbus TCP device and the Modbus RTU network

Tip

You can use the manufacturers' *standard* Modbus settings in the drop-down menu associated with the Modbus protocol

The **Save** button allows you to save the serial connection configuration immediately

4.3.5 Equipment Definition File Library

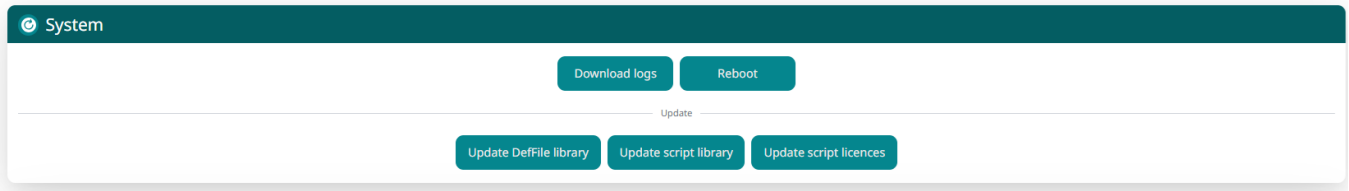
The definition file library allows you to:

- to import definition files
- to delete them
- to edit them
- to export them
- to update them from the Webdyn server

Library Update

From the web server

The first step is to update the internal library with the library available on the Webdyn server. This action is available in the **System>Action** tab:



Update DefFile Library

From the Webdyn website

You can also download the library from: Link

Via server or SMS

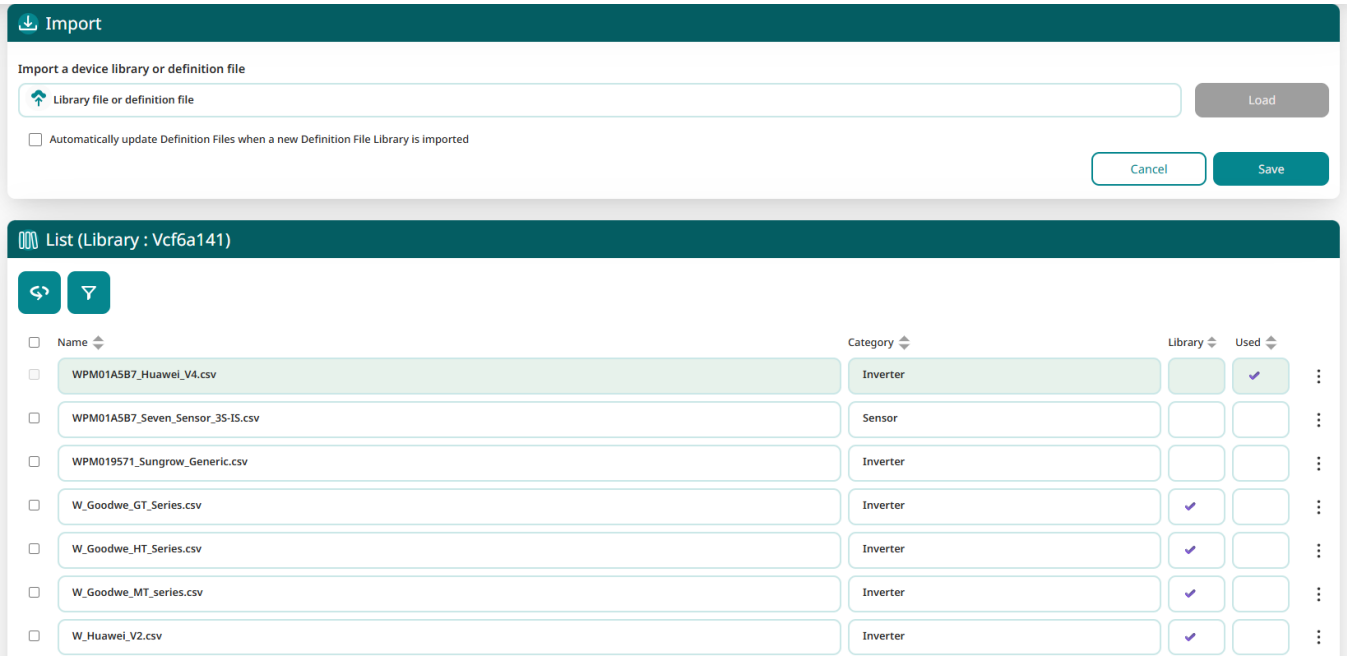
You can update the library from Server 1 or with SMS using the following command:

`updateLibrary`

Please refer to the appendix on External Commands.

Library Management

The library version is available through the interface.



def file library page

The list of definition files available on the gateway is displayed in the list along with the following information:

- **Name**: definition file name
- **Category**: equipment category read from the definition file
- **Library**: File source
  - If checked: The file comes from the library but is not in use
  - Otherwise: the file was imported as a single unit
- **Used**: Definition file currently in use by a device

> **Note**
>
> The file defining the gateway's internal I/O and the Modbus TCP file are not included in this list

> **Note**
>
> When a device uses a library definition file, the file is automatically renamed. The library file `W_<Manufacturer>_<Model>.csv` is then renamed to `<UID>_<Manufacturer>_<Model>.csv`.
>
> If the equipment is deleted, the definition file reverts to the one in the library.

Import

You can manually import either:

- a definition file—in `.csv` format
- a bookstore - in `.libw` format

Checking **Automatically update Definition files when a new Definition File library is imported** updates the definition files currently in use as soon as the library is imported (by file or action). Otherwise, the definition file is retained.

> **Warning**
>
> Any changes made to the definition files will then be lost

> **Note**
>
> If the imported file is invalid, the gateway will display the reason.

Additional Actions

For each definition file, additional actions are available at the end of the line:

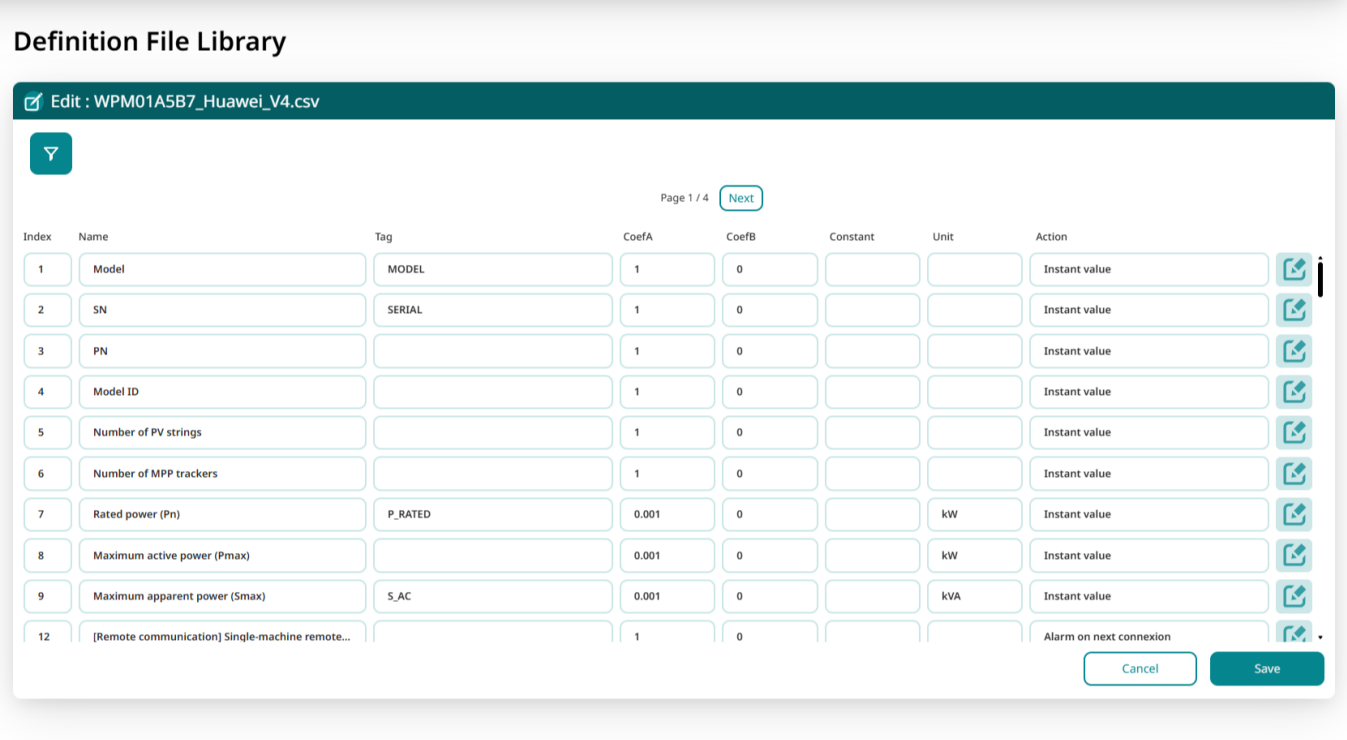
Action	Description	Status: File in the library	Status: File in use	Status: Manually added, unused file
Info	Lists the devices that use the file	✗	✓	✓
Category	Change the category	✗	✓	✓
Edit	Editing the file	✗	✓	✓
Export	Download the file	✓	✓	✓
Restore	Restores the file from the internal library	✗	depends on the source	depends on the source
Delete	Deletes the file	✓	✗	✓

Editing the definition file

You can edit a definition file directly from the web server. The file must be either:

- in use OR
- added manually

> **Note**
>
> It is not possible to edit a file from the internal library.



def file edit page

The definition file is displayed as a table containing certain elements of the definition file, as described in the corresponding section:

- **Index**: Unique index
- **Name**: Unique name
- **Tag**: Unique tag
- **CoefA**: Coefficient
- **CoefB**: Offset
- **Constant**: constant value associated with the action **constant value**
- **Unit**: Unit
- **Action**: Action/Type

> **Note**
>
> With the **constant value** action, only a record can be used to edit the **Constant** column.

4.3.6 Script Library

> **Note**
>
> The script library has been available since FW V5.2

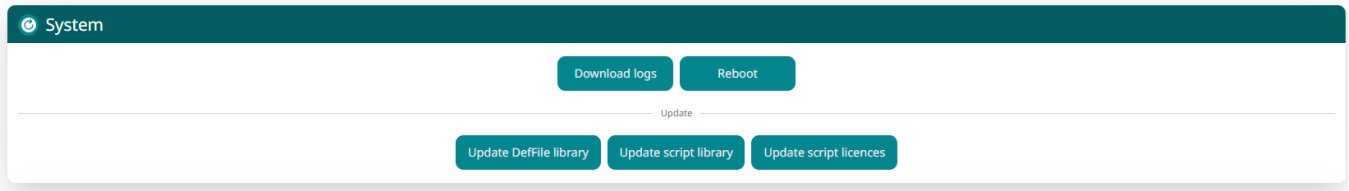
The script library allows you to:

- to import unit scripts
- to import a script library
- to import a license file
- to update the script library
- to enable and configure the scripts

Library Update

From the web server

The first step is to update the internal library with the library available on the Webdyn server. This action is available in the **System>Action** tab:



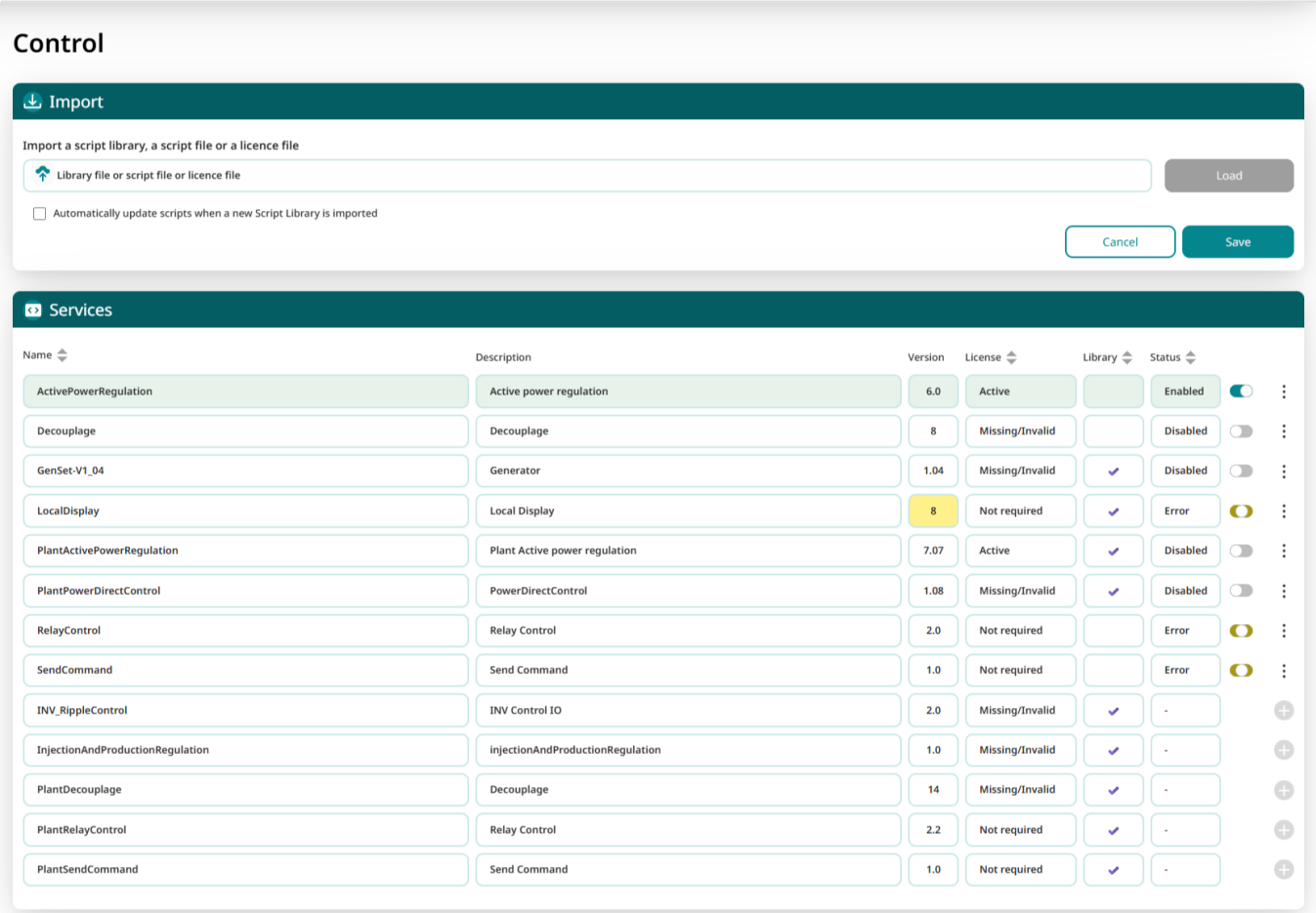
Update DefFile Library

From the Webdyn website

You can also download the library from: Link

Library Management

In the **Controls** tab, you can view the script library and the scripts installed individually:



control page

Import

You can import manually:

- A script library in the `.slibw` format
- A single script in unencrypted format `.lua` or encrypted format `.luaw`
- An activation license in the `.ini` format

Checking **Automatically update scripts when a Script library is imported** allows the library's scripts to be updated automatically, whether they are currently in use or not.

> **Warning**
>
> When the script is updated, you need to reconfigure it

List of Scripts

In the **Services** section, the list of scripts appears in two sections:

- Scripts in the gateway's internal library (3-dot icon)
- Scripts available in the Webdyn library (plus icon)

The information displayed is:

- **Name**: Script name
- **Description**: Additional Information
- **Version**: script version (installed or in the library)
- **License**: License status for the script
- **Library**: Script that is part of the remote library managed by Webdyn
- **Status**: Script status

Bachelor's Degree	Description
Not Required	No license is required
Missing/Invalid	License required but not available for the gateway
Active	Required license activated

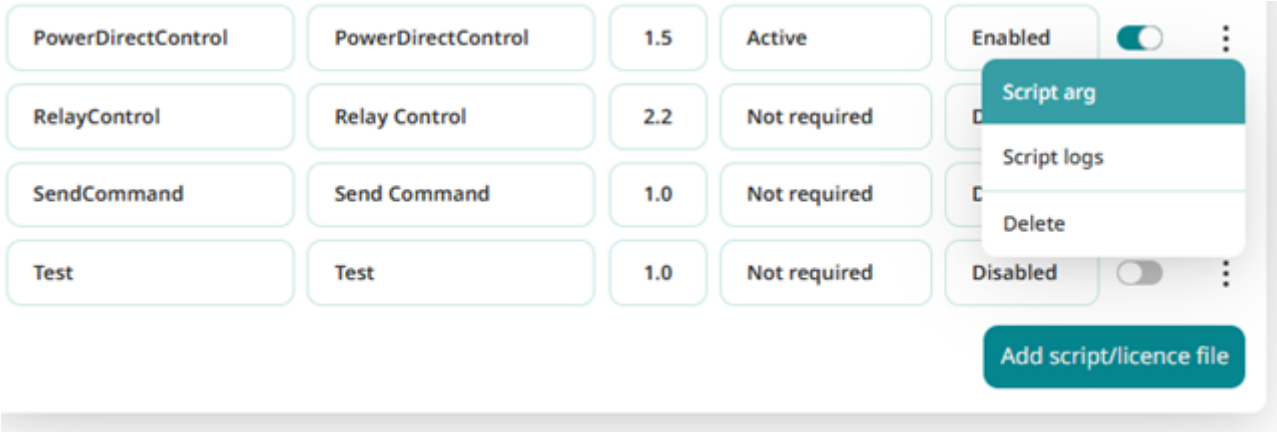
Status	Description
Disabled	The script has stopped
Error	An error is preventing execution—see the logs
Enabled	The script is currently running

Scripts that have a newer version in the remote library have an jaune entry in the **Version** column

Actions

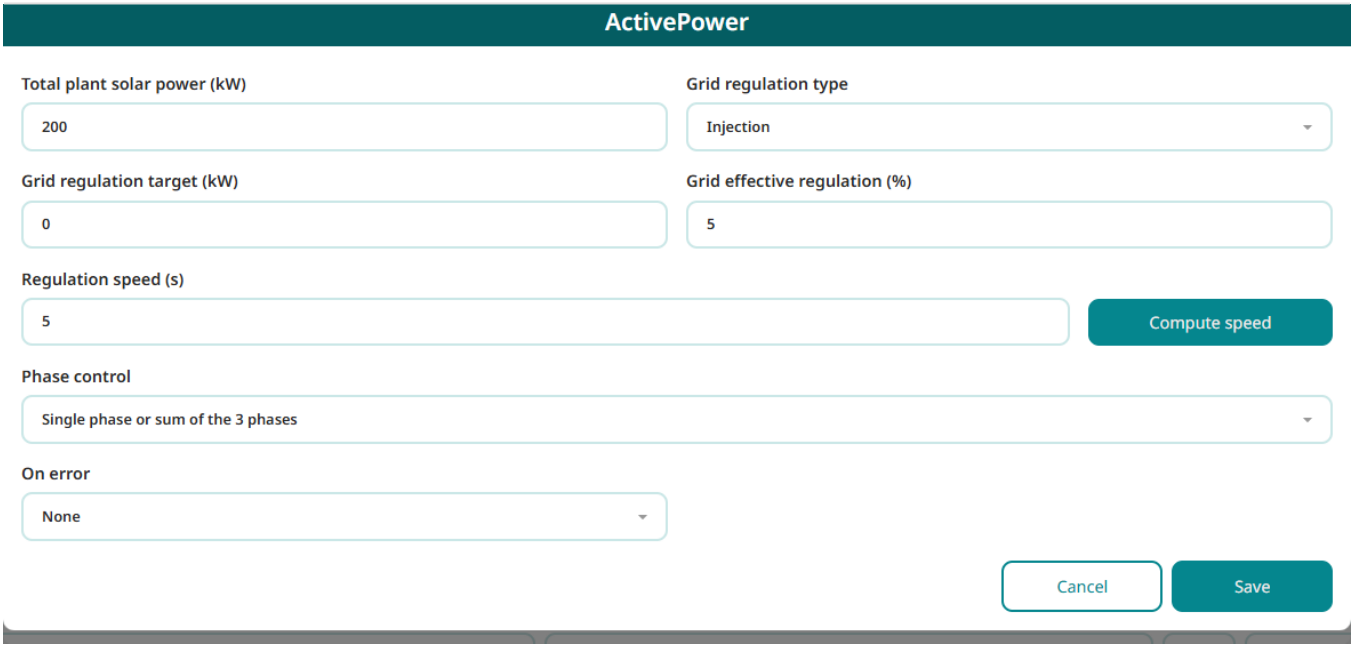
Settings

In the additional actions in the options menu, you can configure the scripts



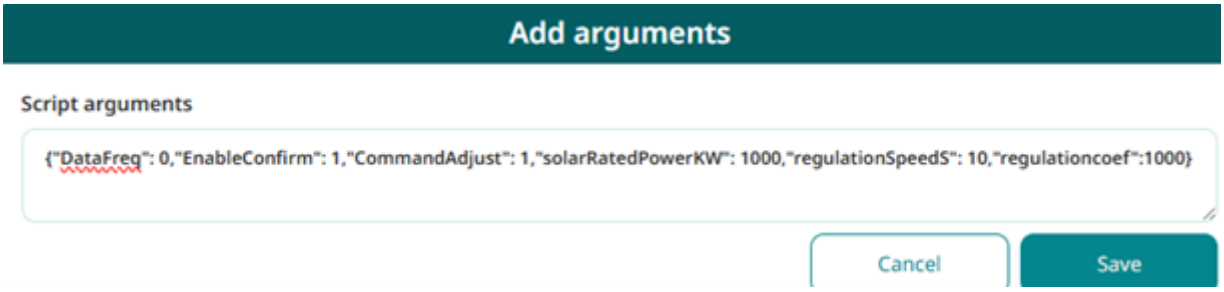
script arguments

Some scripts have a settings page:



script arguments 3

Others simply provide a line in the format `json` :



script arguments 2

Scripts can also be configured:

- From the main server
- From the secondary server (MQTT)

Activation

The script runs with the end-of-line slider:

script activation

Scripts can also be activated by:

- Text Message
- Primary Server

Scripts Logs

You can monitor the script's execution in its log, which is available in the options (**3 points** menu)

script logs

Scripts deletion

The script can be removed from the gateway using the **Delete** action in the options (**3 points** menu)

Scripts from the library will then be available again and displayed as being in the library (plus icon)

4.4 Commissioning by the main server

> **Note**
>
> **Prerequisites**
>
> To initiate commissioning via the main server, the gateway has the following:
>
> - working Internet connectivity
> - a configuration for accessing the main remote server (FTP/SFTP/WebDAV)
> - The remote server exposes a directory structure that matches the one configured on the gateway.

4.4.1 Configuration Files

This operating mode requires an FTP, SFTP, or WebDAV server that is accessible by the gateway. This server can be accessed using an address, a login ID, and a password.

The server can be hosted on any type of server (Windows, Linux, etc.). The important thing is that its address be accessible to the gateway using the network interface that will be configured (Ethernet or modem). It is preferable to use an SFTP or WebDAV server because they offer additional layers of security compared to a traditional FTP server. Aside from that, the various types of servers function the same way. SFTP is based on a username and password.

The gateway's configuration files are located in the specified configuration directory, which is `/CONFIG` by default.

The files are as follows:

- **<UID>_config.ini**: This file contains the following configuration settings:
  - Configuring the Server Directory Structure
  - Configuring the server type: FTP, SFTP, WebDAV, MQTT, etc.
  - NTP Configuration
  - Gateway Name and Description
- **<UID>_var.ini**: This file contains the configuration settings for the gateway's connection scheduling.
- **<UID>_daq.csv**: This file contains the gateway's interface configuration and the list of monitored devices:
  - **Modem Configuration**:
    - PIN

        - APN
        - Username/password/authentication type
    - **Ethernet**:
        - IP Address
        - Footbridge
        - DNS
    - **Serial** :
        - Speed
        - Parity
        - Data bits
        - Protocol type used: Modbus, etc.
    - **Connected Devices** :
        - Index
        - Last Name
        - Interface
        - Address
        - Definition File
- **<UID>_licence.ini**: This file contains the licenses for Webdyn Lua scripts (.luaw)
- **<UID>_scl.ini**: This file contains the configuration settings for the scripts installed on the gateway

With `<UID>`, the gateway ID.

Descriptions of the configuration files are available in the appendices.

Establishing the Connection

The configuration will be synchronized between the gateway and the server the next time a connection is established. A connection to the server can be initiated in three ways:

- Initiated by the gateway:
    - Scheduler: configured by the web server or `<UID>_var.ini`
- User-initiated:
    - Web server: **Connect** button
    - SMS: **Connect**

Behavior Upon First Connection

Case 1 - No previous file on the server

→ Initialization from the gateway

- The gateway's internal files are sent to the server

Cas 2 - Existing files on the server

→ Server Priority

- Remote files replace the gateway's internal files
- Allows:
    - Pre-configuration
    - Replace if `<UID>` has been modified to match the gateway to be replaced

> **Warning**
>
> It is strongly discouraged to have gateways with the same `<UID>` in operation

Behavior at Rated Conditions

Timestamp synchronization

For each file:

Case	Action
Newer server file	Uploaded to the gateway
Latest gateway file	Sent to the server

4.4.2 Useful Commands

Start a SunSpec Automatic Discovery

Using the `DiscoverDevices` function, you can search for and add SunSpec devices:

`discoverdevices=<protocol>:<maxDevices>:<interface>:<timeout>:<port>`

To run this command from the server, you must place a JSON file in the command directory (by default `/CMD`).

Here is an example of a command file for the `DiscoverDevices` function:

```
{
        "rpcName":"sunpm.discoverdevices",
        "parameters":{
                "protocol":"sunspec",
                "maxDevices":5,
                "interface":"ethernet",
                "timeout":1000,
                "port":6000
        },
        "callerId":"1"
}
```

This example will initiate a search for up to 5 SunSpec devices using Modbus TCP.

Retrieve the gateway's mobile number

When the gateway is equipped with a SIM card, you can retrieve the mobile number using the following command:

`status;<number phone>`

Here is an example of a JSON file:

```
{
        "rpcName": "sunpm.status",
        "parameters": {
                "number": "+33700000000"
        },
        "callerId": "1"
}
```

Here, the gateway will send a response to the number +33700000000:

```
type=WebdynSunPM 4G
idSite=WPM01194D
FW=5.0.2.41619
ModeServ1=modem
apn=tm
ipMobile=10.6.157.235
csq=26,99
network=4G
ip=192.168.1.12
mask=255.255.255.0
gw=
dns=
srv1:
type=ftp
addr=user@ftpserver.com:21
last=12/10 12:53 success
```

This will then allow you to find out the gateway's number.

4.5 Activation via SMS

> **Note**
>
> Starting with version v5.1.0, SMS messages received by the gateway must be encrypted. The encryption method is described in the application note, along with a site web that can be used to encrypt SMS messages.

> **Note**
>
> An appendix listing all the commands is available

Prerequisites

To set up the service via text message, you must first complete the following steps:

- Insert a SIM card with a valid service plan and find out the number
- Set up the PIN if necessary

APN Settings

To allow the gateway to connect to remote servers, you must configure the APN.

SMS Command Format

```
apn=<apn>:<login>:<password>
```

Settings

Parameter	Required	Description
apn	✓	Camera URL
login	Depends on the mobile carrier	username
password	Depends on the mobile carrier	password

SMS Command Example

```
apn=m2mapn:login:webdyn
```

SMS Response Format

- Success: None
- Failure: failure message

Configuring the Primary Server for SFTP

You can then configure the connection to the main server, which will enable remote monitoring.

SMS Command Format

```
sftp=<server>:<login>:<password>:<port>:<interface>
```

Settings

Parameter	Required	Description
server	✓	SFTP Server Address
login	✓	username
password	✓	Password
port	✓	port
interface	✓	`Ethernet` or `Modem`

SMS Command Example

```
sftp=sftp.webdyn.com:login:password:22:modem
```

SMS Response Format

- Success: None
- Failure: failure message

Initiating the first connection to the main server

To check whether the primary server configuration is working, the `connect` command initiates an immediate connection.

SMS Command Format

```
connect=<connexion>
```

Settings

Parameter	Required	Description
server	✗	server ID (`1 = Server 1` , `2 = Server 2`)

SMS Command Example

```
connect=2
```

SMS Response Format

- Success: None
- Failure: failure message

If no text message is received, the connection has been established and the gateway has synchronized its files with those on the main server.

The gateway is now ready to be managed remotely.

4.6 Modbus TCP Slave

4.6.1 Configuration by Primary Server

The Modbus slave is configured by editing the `<UID>_daq.csv` file in the configuration directory (default: `/CONF`) on Server 1 (the primary server).

`ModbusSlave;<enabled>;<port>;<mapping file>`

With:

- **enabled**: Modbus slave status:
  - 0 → Off,
  - 1 → Enabled.
- **port**: TCP port used by the server. By default, port 502 is used.
- **mapping file**: Name of the definition file describing the mappings between user registers and variables in devices or Lua scripts. The file must be located in the definition file directory (by default, `/DEF`).

> **Note**
>
> If no definition file is specified, only the WebdynSunPM variables are accessible.

Example of an excerpt from a `<UID>_daq.csv` file:

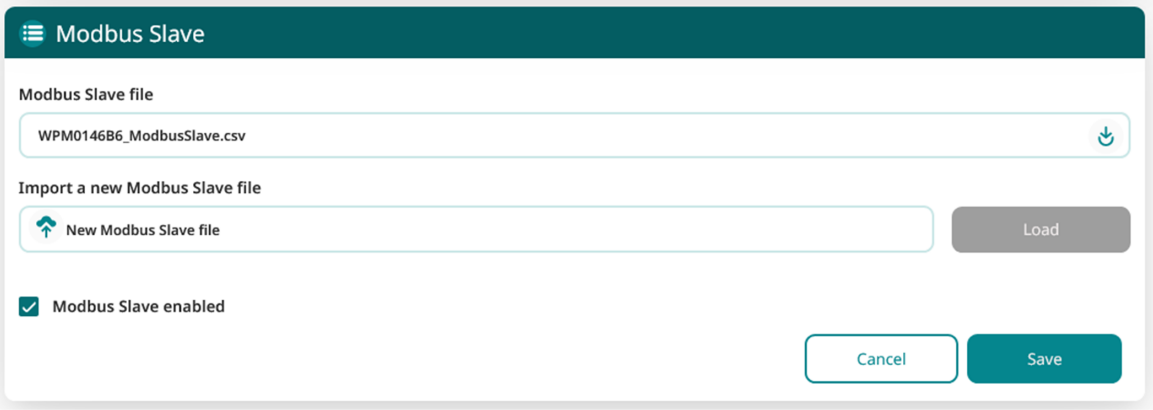
```
...
SERIAL3;9600;8;N;1;2;modbusRTU;0;

ModbusSlave;1;502;WMP0116AB_ModbusSclave.csv
```

```
index;interface;name;address;acqPeriod(s);timeout(ms);serialNumber;parameters;category;model;def
File
...
```

4.6.2 Configuration via the Web Server

The Modbus slave is configured from the **System>Settings** tab:



modbus slave page

4.6.3 Definition File

The file name can be chosen freely and can be changed by the user as desired; the gateway will use the name provided in the **<UID>_daq.csv** file.

In the example below, we retrieve variables from two different inverters and one variable created by a Lua script, as shown in the following list:

1. Read the variable named **Temperature** in U16 format from **INVERTER1** (1 register, or 2 bytes).
2. Read the variable named **E-Total** in F32 format from **INVERTER1** (2 registers, or 4 bytes).
3. Read the variable named **Firmware Version** as a string with a maximum length of 20 bytes from **INVERTER2** (i.e., 10 registers).
4. Read the variable tagged **DisplaySerialNumber** in string format, with a maximum length of 24 bytes, from INVERTER2 (i.e., 12 registers).
5. Read and write the **ValueRegulation** variable of the virtual device in F32 format, created by the Lua script **regulation.lua** (4 registers, or 8 bytes).

Example definition file:

```
ModbusSlave
1;;0;;INVERTER1;Temperature;;;;;1
2;;1_4;;INVERTER1;E-Total;;;;;1
3;;3_20;;INVERTER2;Firmware Version;;;;;1
4;;13_24;;INVERTER2;;DisplaySerialNumber;;;;1
5;;25_8;;regulation;ValueRegulation;;;;;4
```

Warning

Variables are referenced either by their name or by their tag

4.7 Setup via SD Card

Configuration via a microSD card is similar to configuration via the main server.

The only difference is that the gateway does not need a connection to the remote server, since all files will be accessible directly from the inserted SD card.

Also, since the directories on the SD card cannot be configured, the directory structure must follow this format:

- /CONFIG
- /ALARM
- /LOG
- /BIN
- /CERT
- /DATA
- /CMD
- /DEF
- /SCRIPT

If the directories do not exist, they will be created on the SD card the next time a connection is requested. If the gateway is configured to use the SD card and the user performs a test connection, the device will search for any configuration files on the card and use them.

The Micro SD card is recognized and treated as an FTP server by the gateway

> **Note**
>
> Command files (/CMD) are not processed

5 Operation

5.1 SFTP/Webdav Server

5.1.1 File system

Tree Structure

The server's directory structure must match the one configured on the gateway. The gateway must have the required permissions. Below is a list of the files that will be exchanged between the gateway and the server.

File	Description	Rights	Default directory
<UID>_config.ini	Gateway Configuration	RW	/CONFIG
<UID>_daq.csv	Interface Configuration	RW	/CONFIG
<UID>_var.ini	Scheduler Configuration	RW	/CONFIG
<UID>_scl.ini	Script Configuration	RW	/CONFIG
<UID>_licence.ini	License	RW	/CONFIG
<UID>_ <Manufacturer> _<Model>.csv	Definition of Equipment	RW	/DEF
<UID>_ <interface> _<timestamp>.csv.gz	Data Collected	W	/DATA
<UID>*AL*<timestamp>.csv.gz	Alarms Generated	W	/ALARM
<comments>.lua	Script	RW	/SCRIPT
<UID>_cmd.json	Orders	RW	/CMD
<UID>*LOG*<timestamp>.log.gz	Journal	W	/LOG
<UID>*SYSTEME*<timestamp>.tar.gz	Debug Log	W	/LOG
wgapp_<version>.spm	Update	R	/BIN
<comments>.pem	Certificates	RW	/CERT

Starring:

- **<UID>**: Unique identifier for the gateway
- **<timestamp>**: The timestamp format is **AAMMJJ_HHMMSS**
- **<interface>**: name of the relevant interface
- **<comments>**: free-form field for the user
- **<version>**: update version number. Data, alarm, and log files are compressed in Gzip format ".gz **."

> **Note**
>
> All files exchanged between the gateway and the server must be encoded in the standard UTF-8 format.

File Exchange and Synchronization

For an SFTP/FTP server, the **TwoStepsSendingDisabled** configuration parameter allows you to upload files as follows:

1. During the transfer, the file has the extension `.tmp`
2. At the end of the transfer, the extension `.tmp` is removed

This lets you know if a file is currently being transferred.

Sync based from timestamp

For each file:

Case	Action
Newer server file	Uploaded to the gateway
Latest gateway file	Sent to the server

5.1.2 Gateway Configuration

> **Note**
>
> The configuration can only be modified from Server 1

During normal operation, the files currently being used by the gateway are those stored on the server.

The configuration files are located in the configuration directory (default `/CONFIG`):

- `<UID>_config.ini` : Gateway Configuration
- `<UID>_daq.csv` : Configuring Communication Interfaces
- `<UID>_licence.ini` : License
- `<UID>_var.ini` : Server Connection Scheduler
- `<UID>_scl.ini` : Script Configuration

The changes made will be transferred to the gateway the next time you connect.

> **Warning**
>
> Any changes to the configuration files may cause the gateway to malfunction.

You must use the configuration file names as specified above.

After applying the new configuration, the result is recorded in the gateway's log file.

If there is an error in the configuration file—such as an incorrect value—the gateway will ignore it and use the default value if one exists; otherwise, the file will be rejected.

The log file will record the error and the default value that was applied.

Update the definition file library

Using the `updateLibrary` function, you can update the library from the server to get the latest additions and updates:

`updateLibrary`

You must send a JSON file to the command directory (default `/CMD`) as shown in the following example:

```
{
        "rpcName":"sunpm.updateLibrary",
        "callerId":"674"
}
```

Update the license file

When a software license is purchased, the sales department sends the buyer a file named `<UID>_licence.ini`. This file can then be placed in the configuration directory (by default, `/CONFIG`).

The next time you log in, the license file will be downloaded by the gateway and will activate the corresponding script.

Next, you'll need to configure it and start it.

5.1.3 Analyzing the collected data

File format

The data is sent to the data files directory (default: `/DATA`). The data files are in the `.csv` format but are compressed to reduce their size. The compression uses the Gzip format `.gz`.

The format of the data file names is as follows:

`<UID>_<interface>_<timestamp>.csv.gz`

With:

- **<UID>**: Gateway ID
- **<interface>**: interface name from the following list:
  - **TIC**: USB-TIC extension for customer teleinformation
  - **IO**: Internal I/Os of the gateway
  - **Modbus**: Modbus devices
- `<timestamp>`: Date in the format **AAMMJJ_HHMMSS**

Examples:

`WPM00C44F_Modbus_210112_105947.csv.gz` `WPM00C44F_IO_210202_084443.csv.gz`
`WPM00C44F_TIC_210202_095243.csv.gz`

As a result, data collected from various devices—including Modbus devices—is stored in the same data file.

> **Note**
>
> One file per interface is generated for each connection to the server

> **Note**
>
> If a connection to the server fails, the gateway will attempt to resubmit the file on the next attempt. There will then be two data files. The files are sent in order from oldest to newest.

> **Warning**
>
> Each device has 50 MB of storage space for uncompressed data. If this limit is reached, no new data will be saved.

Contents

The contents of a data file consist of two parts, which are:

- A header: specific to the device or interface.
- Data: Single Format

The reported values are always raw and must be linked to the equipment definition file.

IO Header (Internal Inputs/Outputs)

The header of the IO data file is as follows:

```
TypeIO;fileDefinitionName
```

With:

- TypeIO: Data type identifier
- fileDefinitionName: name of the associated definition file

Modbus Equipment Header

For each piece of equipment, a header is provided:

```
DEVICEINDEX;IdxDevice_1
Protocol_1;DefinitionFileName_1

<data_IdxDevice_1>
...
DEVICEINDEX;Idx_Device_N
Protocol_N;DefinitionFileName_N

<data_IdxDevice_N>
```

With:

- **IdxDevice_N**: Equipment index N in the configuration file `<UID>_daq.csv`.
- **DefinitionFileName_N**: Definition file associated with equipment N in the configuration file `<UID>_daq.csv`

Data

Each device has its own acquisition frequency. The gateway continuously collects data from the devices, and at each acquisition step, a data point is either stored or not, depending on the configuration of the variable in the definition file associated with the device.

The data type is named **Code Action** and can have the following values:

Code	Description
0	Variable is disabled; it will not appear in the data files
1	Parameter; will not appear in the data files
2	Three fields will be filled in the data files
4	Instantaneous value
6	Instantaneous value; can be retrieved using the **GetData** command
7	Min / Max / Average; can be retrieved using the **GetData** command
8	Alarm (immediate activation)
9	Alarm (postponed until the next login)
10	Constant; does not appear in the data files

The data in the equipment or interface data file is as follows:

Optional header

Depending on the **OptionHeader** setting in the `<UID>_config.ini` configuration file, this line may or may not be included.

```
nbVariableDevice_N;indexVariable_1_Device_N;indexVariable_A_Device_N;EnableAdvancedData
```

With:

- **nbVariableDevice_N**: Total number of variables collected from device N
- **indexVariable_x_Device_N**: Index of variable X for device N
- **EnableAdvancedData**: configuration file parameter `<UID>_config.ini`; if enabled, the number of reads during the acquisition period will be provided.

Acquisition Data

Each line represents an acquisition period for the equipment in question.

```
datetime_1;variable_1_value_1_Device_N;variable_x_value_1_Device_N;nb_refreshes_during_1
datetime_2;variable_1_value_2_Device_N;variable_x_value_2_Device_N;nb_refreshes_during_2
...
datetime_Y;variable_1_value_B_Device_N;variable_A_value_B_Device_N;nb_refreshes_during_Y
```

With:

- **datetime_Y**: Data timestamp at acquisition point Y. For the format, see parameter **EuroDateFormat** in the configuration file `<UID>_config.ini`.
- **variable_A_value_Y_Device_N**: Value of variable A at acquisition point Y and for the action defined in equipment definition file N.
- **nb_refreshes_during_Y**: Number of complete reads during this acquisition period. For I/O, TIC, virtual devices, and the parameter collection file, the number of refreshes is always 0.

> **Note**
>
> If no data was collected during an acquisition period, the variable values will be left blank

Modbus Device Example

Example of a data file with the following parameters:

- Server connection schedule: every 15 minutes
- EnableAdvancedData: enabled
- OptionHeader: enabled
- EuroDateFormat: enabled
- Equipment 1:
  - Acquisition Period: 300 seconds
  - DeviceIndex: 1
  - Protocol: Modbus
  - DefinitionFileName: WPM00C44F_SunSpec_inverter_SMA_Solar_Inverter_9301_ModbusTCP.csv
- Equipment 2:
  - Acquisition Period: 600 seconds
  - DeviceIndex: 2
  - Protocol: Modbus
  - DefinitionFileName: WPM00C44F_Huawei_v4.csv

The two devices have different configurations regarding the variables they collect:

Device 1

Index	Action Code	Collection
1	1	Parameter value
2	0	None
3-11	4	Instantaneous value
12	8	Instantaneous value

Device 2

Index	"Action" Code	Collection
1-2	2	Min, Max, and Average Values

Below is the data file:

`WPM00C44F_Modbus_210205_100135.csv.gz`

```
DEVICEINDEX;1;;;;;;;;;;;
ModbusTCP;WPM00C44F_SunSpec_inverter_SMA_Solar_Inverter_9301_ModbusTCP.csv;;;;;;;;;;
11;1;3;4;5;6;7;8;9;10;11;12;1
21/02/05-09:50:00;32;52;5;102;1;0;1;0;0;0;0;5
21/02/05-09:55:00;29;61;5;99;1;0;1;0;3;0;0;5
21/02/05-10:00:00;35;57;5;108;1;10;0;0;0;0;1;6

DEVICEINDEX;2;;;;;;;;;;;
```

```
ModbusTCP;WPM00C44F_Huawei_v4.csv;;;;;;;;;;;
7;1(min);1(max);1(moy);2(min);2(max);2(moy);1;;;;;
21/02/05-09:50:00;16;32;26.00;52;58;51.00;12;;;;;
21/02/05-10:00:00;4;6;5.50;102;105;103.00;12;;;;;
```

Internal I/Os Example

Example of an IO data file with a sampling interval of every 10 seconds.

The parameters in the definition file are as follows:

Inputs/Outputs	Action Code	Collection
1	4	Instantaneous value
2	0	None
3	4	Instantaneous value
4	4	Instantaneous value
5	4	Instantaneous value
6	2	Min, Max, and Average Values
7	8	Current value + alarm on value change
8	4	Instantaneous value

The I/O data file:

`WPM00C44F_IO_210202_154221.csv.gz`

```
TypeIO;WPM00C44F_IO.csv;;;;;;;;;
9;1;3;4;5;6(min);6(max);6(moy);7;8;9
21/02/02-15:41:10;0;0;5;1;130;170;150;5;0;0
21/02/02-15:41:20;0;1;2;2;130;170;150;3;0;0
21/02/02-15:41:30;1;0;3;1;120;160;140;2;0;0
21/02/02-15:41:40;1;0;6;5;120;170;140;3;1;0
21/02/02-15:41:50;0;0;6;4;130;180;150;5;0;0
21/02/02-15:42:00;0;0;6;5;130;200;160;6;0;0
```

5.1.4 Using Alarms

File Format and Synchronization

Alarms are reported as files in the `.csv` format and compressed using Gzip `.gz`.

They are sent to the alarm directory (by default `/ALARM`) on the remote servers.

There are several types of alarms:

Source: Alarm	Triggering	Message
GATEWAY	Starting the gateway	Power ON
	Gateway Shutdown (>10 mins)	Power OFF
	TIC accessory removed	Incidental TIC Loss
	TIC accessory reconnected	Return of TIC Accessories
IO	Alarm variable that has changed	Definition File Name + Index + Value
MODBUS	Alarm variable that has changed	Definition File Name + Index + Value

The alarm file on the server follows this format:

`<UID>_AL_<timestamp>.csv.gz`

With :

- **<UID>**: Gateway ID.
- **<timestamp>**: The timestamp format is **AAMMJJ_HHMMSS**

Depending on the alarm settings, the file can be sent:

- **On Trigger**: As soon as the event occurs
- **On schedule**: The next time you connect to the server

Alarms Generated by the Equipment

`GATEWAY`-type alarms are generated directly by the gateway and are not linked to any device variable.

They are triggered as soon as a state change is detected and cause an immediate attempt to send data to the configured server.

The format of the GATEWAY-type alarm file is as follows:

```
datetime_1;GATEWAY;info_1
datetime_2;GATEWAY;info_2
...
datetime_Y;GATEWAY;info_X
```

With:

- **datetime_Y**: Data timestamp at acquisition point Y. For the format, see parameter **EuroDateFormat** in the configuration file `<UID>_config.ini`.
- **info_X**: Alarm message

IO/Modbus Alarms

The file format for IO- and Modbus-type alarms is as follows:

```
datetime_1;AlarmSource_1;fileDefinitionName_1;nameEquipment_1;indexVariable_1;value_1
datetime_2;AlarmSource_2;fileDefinitionName_2;nameEquipment_2indexVariable_2;value_2
...
datetime_Y;AlarmSource_X;fileDefinitionName_N;nameEquipment_N;indexVariable_A;variable_A_value_B
```

With:

- **datetime_Y**: Data timestamp at acquisition point Y. For the format, see parameter **EuroDateFormat** in the configuration file `<UID>_config.ini`.
- **AlarmSource_1**: Alarm source (IO or Modbus)
- **fileDefinitionName_N**: Name of the definition file associated with the equipment in the configuration file `<UID>_daq.csv`.
- **nameEquipment_N**: Name of the device in the configuration file `<UID>_daq.csv`
- **indexVariable_A**: Index of variable A in alarm
- **variable_A_value_B**: Value B of Variable A

> **Note**
>
> The values in the alarm file are raw. The coefficient, offset, and unit from the definition file are not applied.

Alarms file example

```
21/02/12-07:00:19;Modbus;Modbus_DELTA_M88H-COM1.20-Sunspec.csv;OndA;48;0.000000
21/02/12-07:00:49;Modbus;Modbus_DELTA_M88H-COM1.20-Sunspec.csv;OndA;48;3.000000
```

The Case of Virtual Equipment

Virtual devices can also trigger alarms when the value of a variable configured as an alarm changes.

A virtual device is not queried directly via a physical interface: its variables are populated by the application, particularly through scripts. When an alarm variable changes, the alarm is treated as a device/variable alarm and uses the same format as **IO/Modbus** alarms.

In the alarm file, there is no specific source `VIRTUAL_DEVICE`. For exports in remote server format, virtual devices are associated with the event `MODBUS`.

So the format remains:

```
datetime;MODBUS;fileDefinitionName;nameEquipment;indexVariable;value
```

With :

- fileDefinitionName: Name of the virtual device definition file.
- nameEquipment: Name of the virtual device.
- indexVariable: Index of the alarm variable in the definition file.
- value: the variable's raw value at the time of the change.

> **Note**
>
> As with other variable alarms, the values are raw: the coefficient, offset, and unit are not applied.

5.1.5 Send Orders

Command files allow you to perform tasks remotely on the gateway. This makes it possible to instruct the gateway to perform a device scan or retrieve its configuration.

From the command directory (default `/CMD`), the gateway will process JSON files the next time it connects.

Details on the available commands are covered in their respective chapters.

5.1.6 Using Scripts

> **Note**
>
> For more details, please refer to the LUA user manual and the application notes for the relevant scripts.

List of Available Scripts

The `<UID>_scl.ini` file, located in the configuration folder (by default `/CONFIG`), lists all scripts, their activation status, and their settings.

> **Note**
>
> Scripts available in the library but not added to the gateway are not listed

Add scripts

It is possible to add scripts that are not available in the Webdyn library. These scripts can be of three different types:

- Script in `lua` format: unencrypted
- Script in `luaw` format: Webdyn encrypted script
- Script in `luax` format: user-encrypted script

When one of these script formats is added to the scripts folder (by default `/SCRIPT`), the gateway will retrieve it the next time it connects.

You will then be able to run the script remotely with the `startScript` and `stopScript` commands.

> **Note**
>
> The scripts in the library do not appear in this folder even though they are configured on the gateway

Configure and Run a Script

On the server, the `<uid>_scl.ini` file allows you to configure and activate installed scripts; it is located in the configuration directory (by default, `/CONFIG`).

> **Warning**
>
> It is essential that the gateway have a license for certain scripts

Here is an example file:

```
SCRIPT_Args[6]={"VD_DisplayFreq": 0,"EnableConfirm"=1, "CommandAdjust"=1, "solarRatedPowerkW":
1000, "regulationSpeedS": 5}
SCRIPT_Enable[6]=0
SCRIPT_File[6]=PowerDirectControl.luaw
```

With:

- **SCRIPT_Enable[n]** = script status
  - 1: enabled
  - 0: disabled
- **SCRIPT_File[n]** = script file name
- **SCRIPT_Args[n]** = script parameters in the format `json`

> **Note**
>
> The parameters for the script are listed in the dedicated application note

Running a script

Some scripts can be run remotely from the main server. To do this, you must place a file named `<UID>_CMD.json` in the command directory (by default, `/CMD`). For example:

```
[{
  "rpcName": "PowerDirectControl.SetActivePowerPct",
  "parameters": 50,
  "callerId": "1"
}]
```

The gateway will receive this command the next time it connects.

> **Note**
>
> Please refer to the script's user guide for details on remote commands.

5.2 MQTT

5.2.1 Using the MQTT Service

An (unsecured) MQTT server and an MQTTS server work the same way. However, it is preferable to use an MQTTS server, which includes built-in security layers, rather than an MQTT server.

The description in this chapter applies equally to all types of MQTT servers.

Configuration

The MQTT server is configured using the following settings:

- **Address**: IP address or URL.
- **Port**: By default, 1883 for MQTT and 8883 for MQTTS.
- **Identifier**: Depends on the type of MQTT server selected. Authentication can be performed either using a username and password or using certificates and a key that must be imported into the gateway.

- **Applicative indentifier**: A unique server identifier that allows the server to have its own application space.
- **Topics**: Names of the information channels on the server for storing data and alarms.

From an MQTT/MQTTS server, the gateway supports the following actions:

- Data submission,
- Alarm Log,
- Order Receipt,
- Update the gateway.

The following actions are not supported from an MQTT/MQTTS server:

- Configure the gateway,
- Create, add, or edit a definition file,
- Create, add, or edit a script,
- Add or replace a certificate,
- Submit the logs.

Server Directory Tree

An MQTT/MQTTS server has data channels called **topics**. You must specify the same topic name on both the MQTT/MQTTS server and the gateway. A topic must be specified for the data so that the gateway can publish that data.

If you want to send alerts to the MQTT/MQTTS server, you must specify a topic for the alerts.

If you want to receive commands from the MQTT/MQTTS server, you must specify one topic for the commands and another for the command results. How it works:

The gateway connects and sends data to the MQTT/MQTTS server at each scheduled interval.

If the "Alarm" and/or "Command" fields are filled in, the gateway remains constantly connected to the MQTT/MQTTS server in order to execute actions immediately.

5.2.2 Using Alarms

Format

For the gateway to send alarms to the MQTT/MQTTS server, the "Alarm" topic must be specified. The gateway maintains a constant connection with the MQTT/MQTTS server in order to perform the action immediately.

> **Note**
>
> When an alarm is triggered, no data other than the alarm data is stored on the server.

There are several types of alarms:

Source: Alarm	Triggering	Message
GATEWAY	Starting the gateway	Power ON
	Gateway Shutdown (>10 mins)	Power OFF
	ICT accessory removed	Incidental ICT Loss
	ICT accessory reconnected	Return of ICT Accessories
IO	Alarm variable that has changed	Definition File Name + Index + Value
MODBUS	Alarm variable that has changed	Definition File Name + Index + Value

The format of the alarm data in the `JSON` format is as follows:

```
{
        "__MetaData__":{
                "id":"concentrator_id"
        },
        "alarms":[
        {
                "defName":"defName_alarms",
                "deviceName":"deviceName_alarms",
                "source":"source_alarms",
                "value":"value_alarms",
                "variableIndex":"variableIndex_alarms",
                "date":"YY/MM/DD-hh:mm:ss",
                "timestamp":"value_timestamp_alarms"
        }],
        "alarmsDevice":[
        {
                "type":"type_alarmsDevice",
                "info":"info_alarmsDevice",
                "date":"YY/MM/DD-hh:mm:ss",
                "timestamp":"value_timestamp_alarmsDevice"
        }]
}
```

With :

- **alarms**: Alarm originating from an I/O or Modbus source.
- **alarmsDevice**: Alarm originating from GATEWAY.
- **defName_alarms**: Name of the equipment or interface definition file that triggered the alarm
- **deviceName_alarms**: Name of the device or interface that triggered the alarm
- **source_alarms**: Alarm source (IO or MODBUS)
- **value_alarms**: Value of the variable that triggered the alarm.
- **variableIndex_alarms**: Index of the variable that triggered the alarm.
- **type_alarmsDevice**: Alarm type (GATEWAY).
- **info_alarmsDevice**: Alarm information (see the **Message** column in the table above)
- **date**: Alarm timestamp in UTC+tz (parameter **NTP_TimeZone** in the `<UID>_config.ini` file). In the format YY/MM/DD-hh:mm:ss
- **timestamp**: Alarm timestamp in UTC, in milliseconds (Epoch)
- **MetaData**: Metadata about the context of the equipment listed in the file. Currently, the only information transmitted is the WebdynSunPM identifier in the “id” field.

Examples

Example with an alarm from a device


```
{
        "__MetaData__":{
                "id":"WPxxxxxx"
        },
        "alarms":[
        {
                "defName":"WPM00C44F_SunSpec_inverter_SMA_Solar_Inverter_9301_modbusTCP.csv",
                "deviceName":"modbusTCP",
                "source":"MODBUS",
                "value":"106",
                "variableIndex":3,
                "date":"21/02/05-09:50:00",
                "timestamp":1612515000000
        }],
        "alarmsDevice":null
}
```


Example of an alarm from the gateway


```
{
        "__MetaData__":{
                "id":"WPxxxxxx"
        },
        "alarms":null,
        "alarmsDevice":[
        {
                "type":"POWER OFF",
                "info":"GATEWAY",
                "date":"21/02/05-09:50:00",
                "timestamp":1612515000000
        }
}
```


5.2.3 Leveraging Data

Format

The data format is the same regardless of the device or interface.

The reported values are interpreted using the A and B coefficients defined for each variable in the equipment or interface definition file.

The data in the equipment or interface data file is in the following JSON format:


```
{
        "__MetaData__":{
                "id":"concentrator_id"
        },
        "DAQ_Name_eqp_1":[
        {
                "DEF_Name_var_1_eqp_1":"var_1_value_1_eqp_1",
                "DEF_Name_var_2_eqp_1":"var_2_value_1_eqp_1",
                "DEF_Name_var_X_eqp_1":"var_X_value_1_eqp_1",
                "date":"YY/MM/DD-hh:mm:ss",
```

```
                "timestamp":"value_timestamp_1_eqp_1",
                "nbOfRefreshes":"value_nbOfRefreshes_1_eqp_1"
        },
        {
                "DEF_Name_var_1_eqp_1":"var_1_value_Z_eqp_1",
                "DEF_Name_var_2_eqp_1":"var_2_value_Z_eqp_1",
                "DEF_Name_var_X_eqp_1":"var_X_value_Z_eqp_1",
                "date":"YY/MM/DD-hh:mm:ss",
                "timestamp":"value_timestamp_Y_eqp_1",
                "nbOfRefreshes":"value_nbOfRefreshes_Y_eqp_1"

        }
        ],
        "DAQ_Name_eqp_N":[
        {
                "DEF_Name_var_1_eqp_N":"var_1_value_1_eqp_N",
                "DEF_Name_var_2_eqp_N":"var_2_value_1_eqp_N",
                "DEF_Name_var_X_eqp_N":"var_X_value_1_eqp_N",
                "date":"YY/MM/DD-hh:mm:ss",
                "timestamp":"value_timestamp_1_eqp_N",
                "nbOfRefreshes":"value_nbOfRefreshes_1_eqp_N"
        },
        {
                "DEF_Name_var_1_eqp_N":"var_1_value_Z_eqp_N",
                "DEF_Name_var_2_eqp_N":"var_2_value_Z_eqp_N",
                "DEF_Name_var_X_eqp_N":"var_X_value_Z_eqp_N",
                "date":"YY/MM/DD-hh:mm:ss",
                "timestamp":"value_timestamp_Y_eqp_N",
                "nbOfRefreshes":"value_nbOfRefreshes_Y_eqp_N"
        }
        ]
}
```


With :

- **DAQ_Name_eqp_N**: Name of device N, `name` from configuration file `<UID>_daq.csv`
- **DEF_Name_var_X_eqp_N**: Name of variable X for device N; the "Name" field of the variable in the definition file
- **var_X_value_Z_eqp_N**: Z-value of variable X for equipment N, collected at acquisition point Y and for the action defined in the equipment N definition file.
- **date**: Data timestamp at acquisition point Y in UTC+timezone (configured in `<UID>_config.ini`) in the format YY/MM/DD-hh:mm:ss
- **timestamp**: Data timestamp at acquisition point Y in UTC+0, in Epoch milliseconds
- **nbOfRefreshes**: Number of complete readings during this acquisition period. The **MQTT_EnableAdvancedData** parameter must be set to 1 only for Modbus devices and inverters.
- **MetaData**: Metadata about the context of the equipment included in the file. Currently, the only information transmitted is the WebdynSunPM identifier in the **id** field.

Examples

Devices (Modbus, inverters)

Example of an equipment data file with a data collection interval of every 10 minutes:

- Configuring the indexes for Device 1 (ModbusTCP):

Index	Action Code	Display
1	1	Parameter value
2	0	None
3-11	4	Instantaneous value
12	8	Current value + alarm on value change

- Configuring the indexes for Device 2 (SMANET):

Index	Action Code	Display
1-2	2	Min, Max, and Average Values

MQTT data files in `json` format:


```
{
        "__MetaData__":{
                "id":"WPxxxxxx"
        },
        "modbusTCP":[
        {
                "var_1":32,
                "var_3":52,
                "var_4":5,
                "var_5":102,
                "var_6":1,
                "var_7":0,
                "var_8":1,
                "var_9":0,
                "var_10":0,
                "var_11":0,
                "var_12":0,
                "date":"21/02/05-09:50:00",
                "timestamp":1612515000000,
                "nbOfRefreshes":12
        },
        {
                "var_1":35,
                "var_3":57,
                "var_4":5,
                "var_5":108,
                "var_6":1,
                "var_7":10,
                "var_8":0,
                "var_9":0,
                "var_10":0,
                "var_11":0,
                "var_12":1,
                "date":"21/02/05-10:00:00",
                "timestamp":1612515600000,
                "nbOfRefreshes":12
        }
        ],
        "SMANET":[
        {
                "var_1":[16,32,26.00],
```

```
                "var_2":[52,58,51.00],
                "date":"21/02/05-09:50:00",
                "timestamp":1612515000000,
                "nbOfRefreshes":2
        },
        {
                "var_1":[4,6,5.50],
                "var_2":[102,105,103.00],
                "date":"21/02/05-10:00:00",
                "timestamp":1612515600000,
                "nbOfRefreshes":2
        }
        ]
}
```


I/Os

Example of an IO data file with a sampling interval of every 10 seconds:

- Input/Output Configuration:

Inputs/Outputs	Action Code	Display
1	4	Instantaneous value
2	0	None
3	4	Instantaneous value
4	4	Instantaneous value
5	4	Instantaneous value
6	2	Min, Max, and Average Values
7	8	Current value + alarm on value change
8	4	Instantaneous value

MQTT data files in `json` format:


```
{
        "__MetaData__":{
                "id":"WPxxxxxx"
        },
        "IO":[
        {
                "var_1":0,
                "var_3":0,
                "var_4":5,
                "var_5":1,
                "var_6":[130,170,150],
                "var_7":5,
                "var_8":0,
                "date":"21/02/05-09:50:00",
                "timestamp":1612515000000,
                "nbOfRefreshes":0
        },
        {
```

```
                "var_1":0,
                "var_3":1,
                "var_4":2,
                "var_5":2,
                "var_6":[120,160,140],
                "var_7":3,
                "var_8":0,
                "date":"21/02/05-10:00:00",
                "timestamp":1612515600000,
                "nbOfRefreshes":0
        }
}
```


5.2.4 Send Orders

Commands can be sent to the gateway via the MQTT/MQTTS server. To do this, the **Command** and **Result** fields must be filled in the gateway configuration.

When a command is published to the **Command** topic on the MQTT/MQTTS server, it is retrieved by the gateway.

The command is executed by the gateway, and the result of the command is published to the **Result** topic.

The commands are described in detail in their respective sections.

The following is an example of a `writeVariable` command that allows you to write to a variable defined in a definition file:

```
{
        "rpcName":"sunpm.writeVariable",
        "parameters":{
                "deviceName":"Huawei1",
                "tagName":"Watts",
                "value":77
        },        "callerId":"1"
}
```

Using MQTT commands is particularly useful for real-time control because, unlike the SFTP server, commands are processed immediately.

Here is an example of real-time control:

```
{
  "rpcName": "PowerDirectControl.SetPFInteger",
  "parameters": -92,
  "callerId": "1"
}
```

This command controls the power factor of the **PowerDirectControl** script.

5.3 Script License Management

Since scripts are one-time-purchase software options, they are encrypted and require the generation and activation of a license.

5.3.1 License Generation

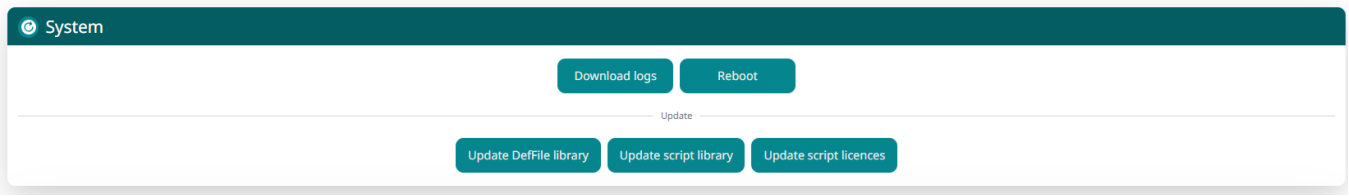
Webdyn handles license generation. You must provide the following information to the sales department:

- **UID / Adresse MAC / Serial Number**: e.g., WPM019571
- **Code article du script**: e.g., PR08-02

Once this information has been submitted and the order has been confirmed, the license is generated `<UID>_licence.ini` and can be sent via Webdyn.

5.3.2 License Activation via the Built-in Web Server

The license is activated in the **System>Actions** tab



actions

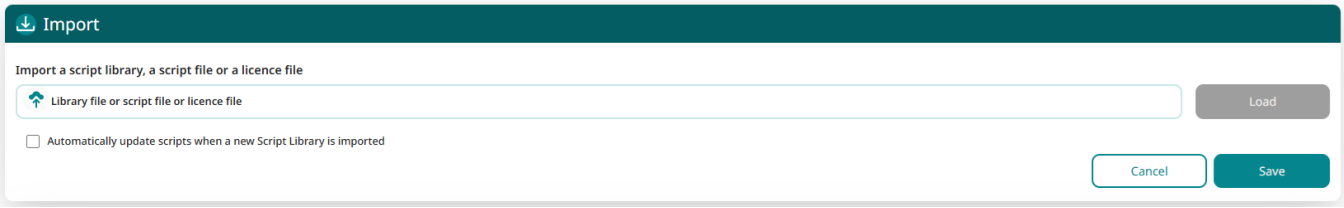
The **Update script Licences** button initiates a connection to the Webdyn server to retrieve the licences associated with the gateway. The **Log** button allows you to see if licenses are available and whether they have been transferred.

Once the licenses have been retrieved, in the **Control** tab, the associated scripts must then specify **Active** instead of **Missing/Inactive**.

script activation

Here, the `Active Power Regulation` script has a license, and the `Decouplage` script does not.

If there is no remote connectivity, you can also load the license from the generated file. In the **Control** tab, you can import the license file.



script import

5.3.3 License Activation by the Primary Server

To activate a script's license, you must place the `<UID>_licence.ini` file in the main server's configuration directory (by default, `/DEF`).

The next time you log in, the license will be downloaded and activated on the gateway.

> **Warning**

If the gateway's name has been changed, you must rename the license file to match the new name.

You do not need to modify the contents of the license file

6 Maintenance

6.1 Firmware Update

The gateway can be updated:

- locally via the web interface
- locally via SD card
- remotely via FTP, SFTP, or WebDAV-HTTPS file upload
- remotely via SMS, MQTT, or FTP/SFTP commands

The latest firmware version is available for download on our website at the following address:

https://www.webdyn.com/support/webdynsunpm/

After downloading, unzip the file (Zip), which contains at least 3 files:

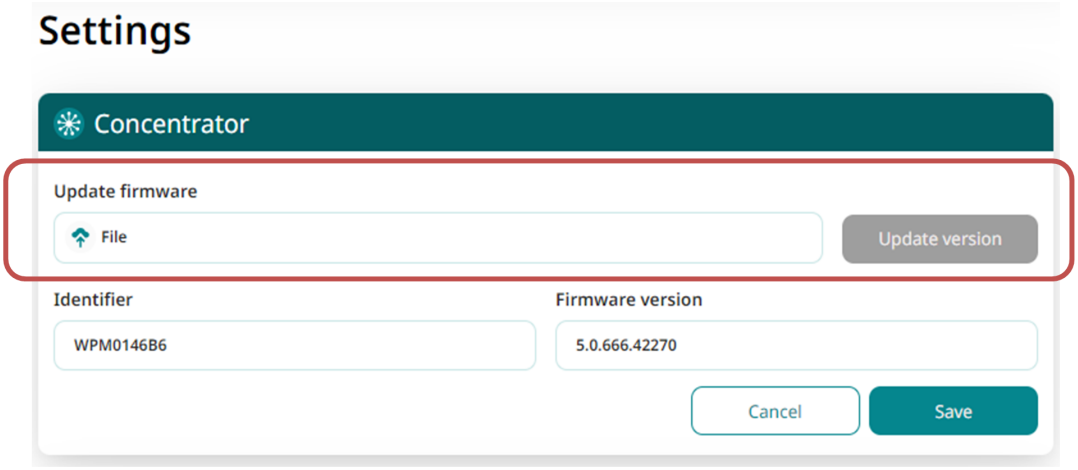
- **wgapp_x.x.x.xxxxx.spm**, which corresponds to the gateway's firmware
- **wgapp_x.x.x.xxxxx.spm.md5**, which includes the firmware checksum
- **release_note_xx.pdf**, which contains the release notes

Note

After each update, it is recommended that you delete the configuration files on the server. This allows the gateway to automatically generate new configuration files that incorporate any changes the next time you connect.

6.1.1 Through the web interface

In the **System>Settings** tab, you can load the `spm` file directly.



firmware update web1

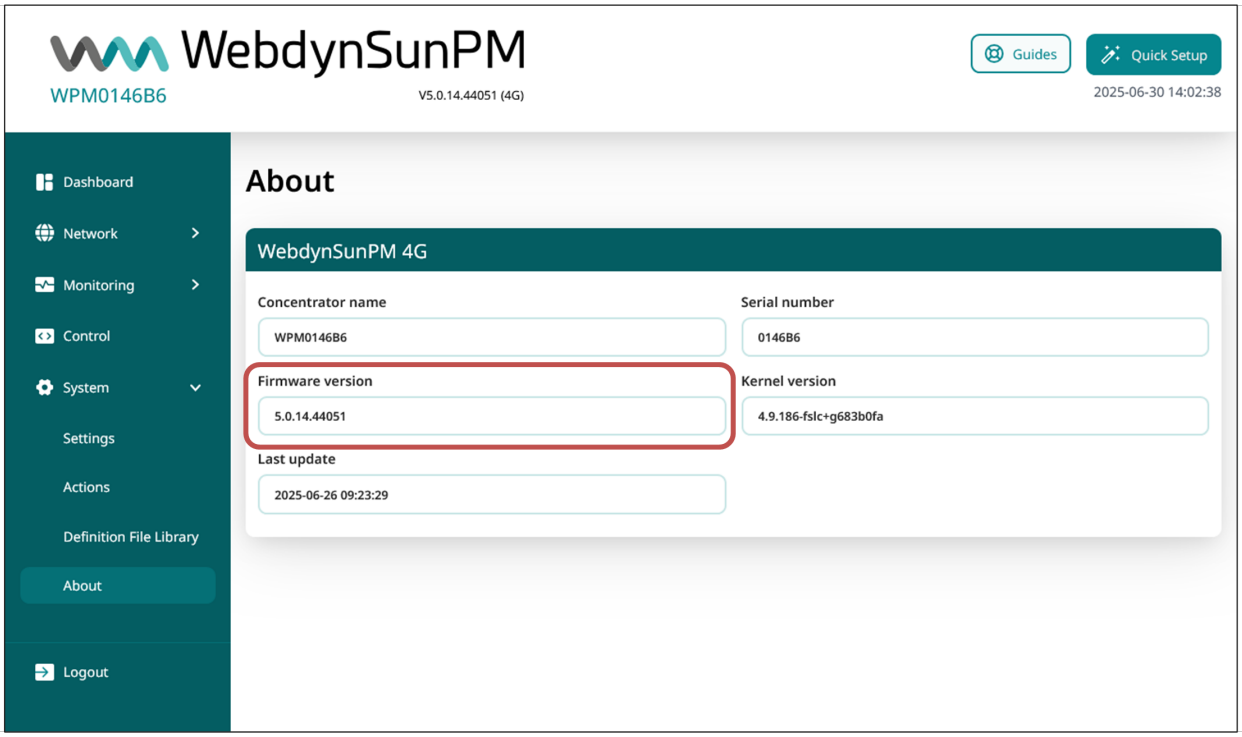
Once it has loaded, you can trigger the update using **Update version**.

firmware update web2

You can view the update's progress from the interface

firmware update web3

When the update is complete, the gateway restarts. You can view the update status on the **System>About** page.



firmware update web4

Tip

After an update, it is recommended that you clear your browser cache by pressing **CTRL-F5**

Tip

After each update, it is recommended that you delete the configuration files on the server. This allows the gateway to automatically generate new configuration files that incorporate any changes the next time you connect.

Note

If an error occurs during the update, the gateway will retain its previously functional firmware. In this case, please carefully repeat the update procedure.

Warning

Do not unplug the gateway, and do not attempt to operate it in any way while the gateway is being updated.

6.1.2 Via SD card

To update via the SD card, follow these steps:

1. Place the **wgapp_x.x.x.xxxxx.spm** file in the `/BIN` directory on the SD card.
2. Edit the **<UID>*config*.ini** file located in the `/CONFIG` directory on the SD card. Enter the name of the file you just placed in the `/BIN` into the variable **BIN_FileName** and enter the checksum specified in the file **wgapp_x.x.x.xxxxx.spm.md5** into the variable **BIN_Checksum**.

The gateway will retrieve its configuration file and then its new firmware from the SD card the next time it connects.

The application of the update can be seen in the log files stored in the `/LOG` directory on the SD card.

After applying the update, you can delete the firmware file and clear the **BIN_FileName** and **BIN_Checksum** variable fields in the **<UID>*config*.ini** configuration file.

If an error occurs during the update, the update will not be retried.

Warning

The update can only be performed this way if the SD card is configured as the primary server.

6.1.3 Via FTP/SFTP/WebDAV

To update via FTP/SFTP/WebDAV, follow these steps:

1. Place the **wgapp_x.x.x.xxxxx.spm** file in the update directory (default: `/BIN`) on the primary server.
2. Edit the **<UID>_config.ini** file located in the configuration directory (by default `/CONFIG`) on the main server. Enter the name of the file you just placed in the `/BIN` into the variable **BIN_FileName** and enter the checksum specified in the file **CheckSumx.x.x.txt** into the variable **BIN_Checksum**.

The gateway will retrieve its configuration file and then its new firmware the next time it connects to the main server.

You can see that the update has been applied in the log files stored in the log directory (by default, `/LOG`) on the main server.

After applying the update, you can delete the firmware file and clear the **BIN_FileName** and **BIN_Checksum** variable fields in the **<UID>*config*.ini** configuration file.

If an error occurs during the update, the update will not be retried.

> **Warning**
>
> Updates via FTP, SFTP, or WebDAV-HTTPS can only be performed by the primary server (Server 1).

6.1.4 Via SMS, MQTT, or FTP/SFTP commands

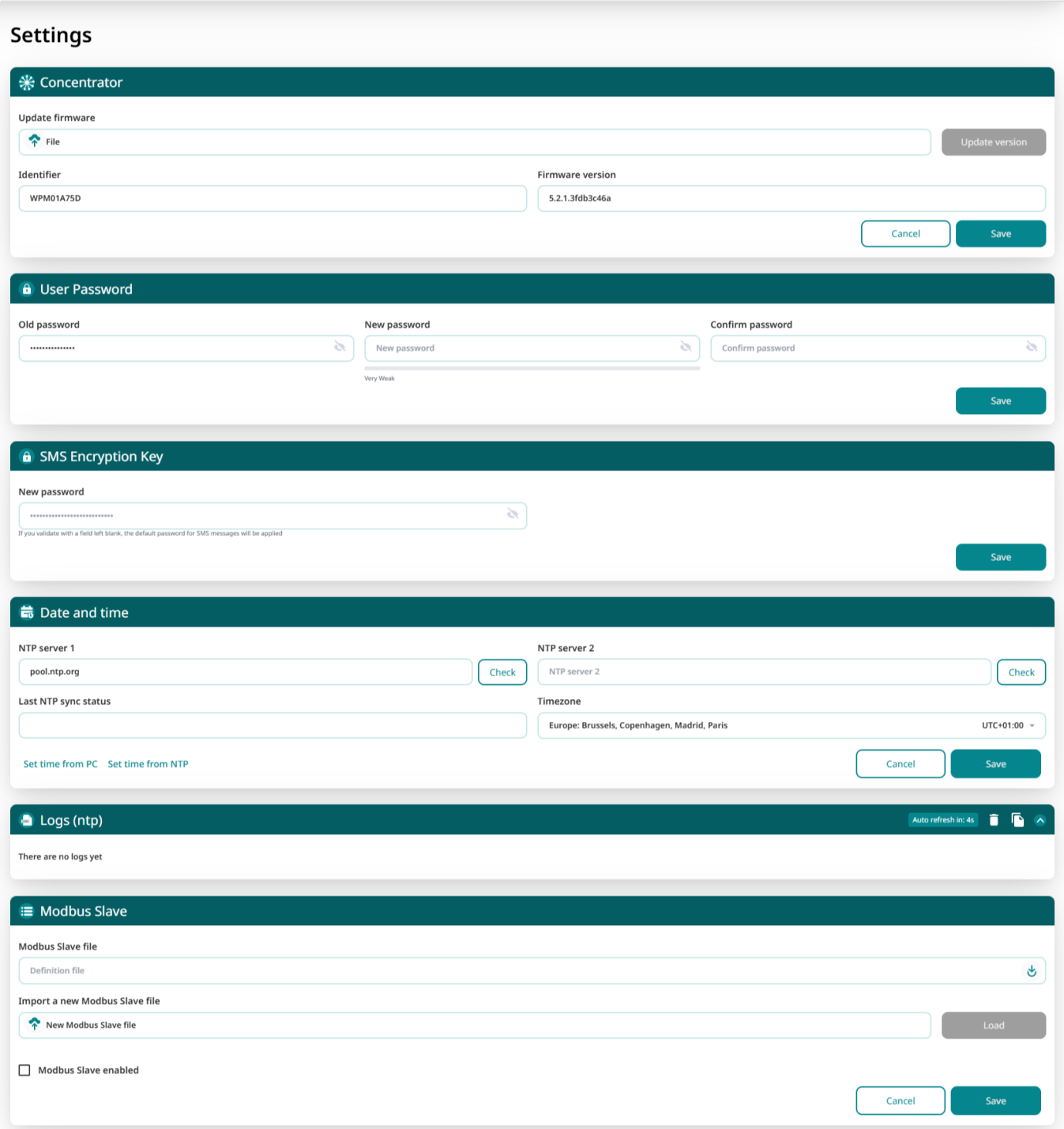
The **updateLibrary** command instructs the gateway to download the firmware file from a URL and then apply it directly.

This command is described in detail in the section on remote commands.

6.2 Changes to System Settings

System settings can be changed from the web server or the main server.

On the web server, the **System>Settings** tab provides access to the edit page:



system settings page

From the main server, you must edit the `<UID>_config.ini` file in the configuration folder (default: `/CONFIG`)—see the corresponding appendix.

File parameter `<UID>_config.ini`	Description
`Concentrator_Identifier`	Username `<UID>`
`WEB_Password`	Web server password
`SMS_Password`	SMS Password
`NTP_Server1`	NTP Server 1
`NTP_Server1`	NTP Server 2
`NTP_TimeZone`	Time zone (see appendix `<UID>_config.ini` for more details)

From the built-in web server, you can test NTP servers directly. You can also set the gateway's time to match that of the browser.

6.3 Restart and Factory Reset

6.3.1 Restart and Shutdown

You can force the gateway to restart from:

- the **POWER** button on the front panel
- Sending an external order `reboot`

The following is a description of the reboot using the **POWER** button:

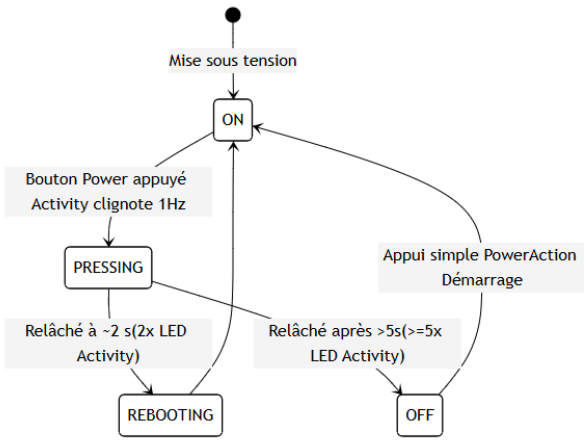


diagram state btn on

When the gateway is operational (see the description of the status LEDs), pressing and holding the POWER button puts the gateway into standby mode:

- If the button is released after two seconds, the gateway restarts.
- If the button is released after 5 seconds, the gateway turns off.

It is also possible to send the external command `reboot` via SMS, a JSON file in the command folder (default: `/CMD`), or via MQTT.

Note

See the appendix on external commands for more details.

Note

It is not possible to turn off the gateway remotely

6.3.2 Return to Factory

The gateway has two functions:

- Resetting Ethernet Interface Settings
- Return to Factory

These actions can be performed using the **RESET** button on the gateway or via an external command `factory`.

The following is a description of the reset using the **RESET** button:

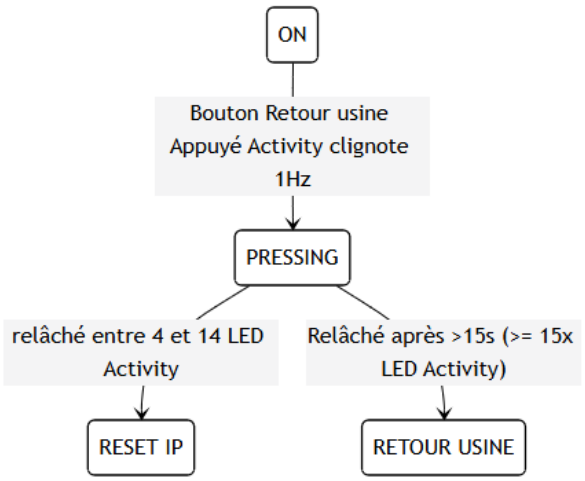


diagram state btn reset

When the gateway is operational, pressing and holding the **RESET** button performs both of the following actions:

- If the button is released between 4 and 14 seconds, the gateway restores the Ethernet connections to their default settings
- If the button is released after 15 seconds, the gateway performs a full factory reset

It is also possible to send the external command `factory` via SMS, a JSON file in the command folder (by default `/CMD`), or via MQTT.

Note

See the appendix on external commands for more details.

> **Note**
>
> Only FACTORY returns are available

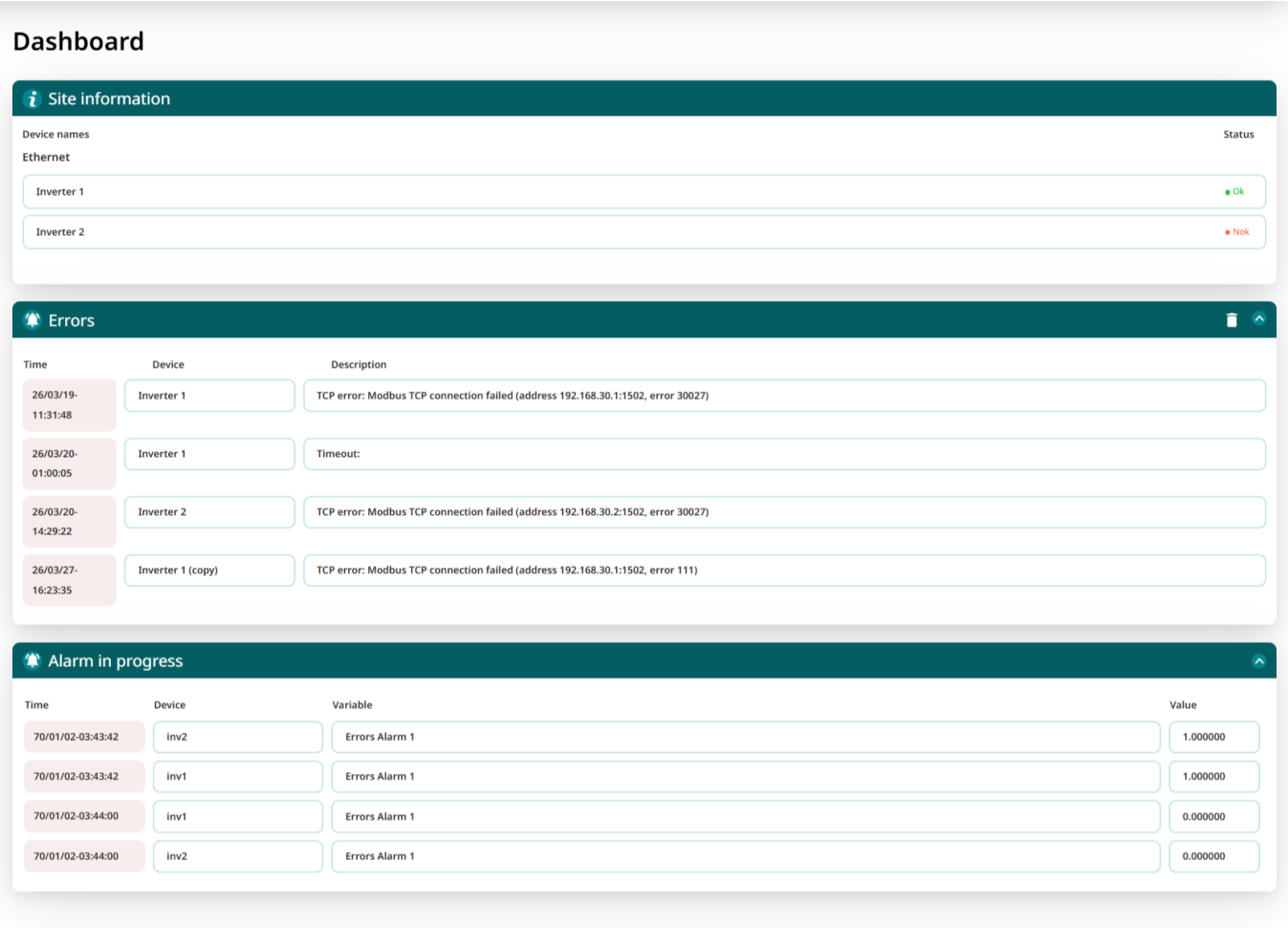
> **Warning**
>
> After a FACTORY RESET, the gateway will no longer be accessible

6.4 Diagnostics

6.4.1 Equipment Communication

Equipment Status

From the web server, the communication status with the devices is displayed in the **Dashboard** tab:



dashboard page

The **Dashboard** tab displays information about the site and the gateway, as well as the overall status of the equipment, current alarms, and communication errors.

At the top of the screen is the site's key information. The status panel displays a list of configured devices with their statuses. The possible statuses for a piece of equipment are:

- **OK**: The device has been found, and the current configuration is functional. (At least one request to the device was successfully responded to; the other requests may have gone unanswered.)
- **Nok**: The device was not found, or the current configuration is not functional. (At least one request from the device has failed, regardless of the status of the other requests.)
- **Unkown**: Unknown equipment status

The **Errors** panel displays a list of the 10 most recent errors detected on the various configured devices. Clicking the trash can icon on the panel resets the list of detected errors.

The **Alarms** panel lists the last fifty alarms.

Communication problem with the equipment

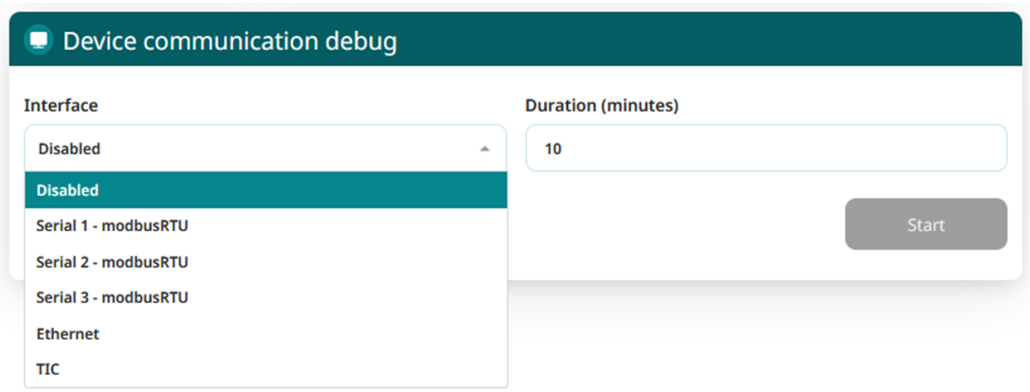
If there is a communication error with equipment, it is often necessary to check the wiring first.

However, if communication with the equipment remains disrupted, a diagnostic tool is available:

- From the web server
- From external orders

Activation from the web server

On the **Monitoring>Device** page, a **Diagnostic** button allows you to configure the tool:



device comms diagnostic page

Select the relevant interface and the duration of the diagnostic test.

When communication traces are selected for a given interface, all communications on that interface will be logged and sent to the server as a log entry timestamped to the nearest millisecond.

> **Note**
>
> This tool generates a very large amount of data, which is why you must specify a time (in minutes) after which the tool will be disabled.

Once complete, a log file is saved on the main server in the logs directory (by default `/LOG`).

Activation from the main server

The diagnostic tool can also be activated remotely from the main server.

`log=<interface>:<duration>`

You must submit a command file named `<UID>_cmd.json` that contains:

```
{
  "rpcName":"sunpm.log",
  "parameters":{
      "interface":<interface>,
      "duration":<duration>
  },
  "callerId":<caller_id>
}
```

With:

- **interface**: The name of the interface on which to start logging: `ethernet`, `tic`, `serial1`, `serial2` or `serial3`
- **duration**: Duration in minutes during which logging will be enabled
- **caller_id**: Request ID for analyzing results

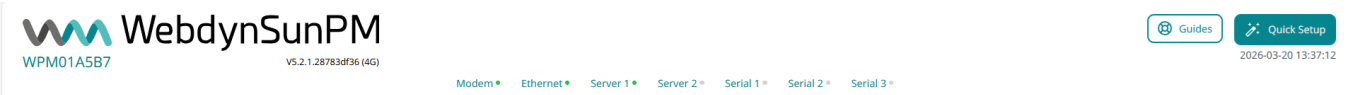
> **Note**
>
> Please refer to the section on external commands for details and a complete explanation of the commands.

Once complete, a log file is saved on the main server in the logs directory (by default `/LOG`).

6.4.2 Remote Communication

From the web server, you can check the status of the remote connection.

You can see the connection status right on the banner:



banner

The status of communication services is tracked with the following statuses:

Modem

- **Connecté**: Remote connection established
- **Erreur**: Any other status
- **Non configuré**: No configuration detected

Server 1 / 2

- **Connecté**: Last successful login
- **Erreur**: Last attempt failed
- **Non configuré**: No configuration detected

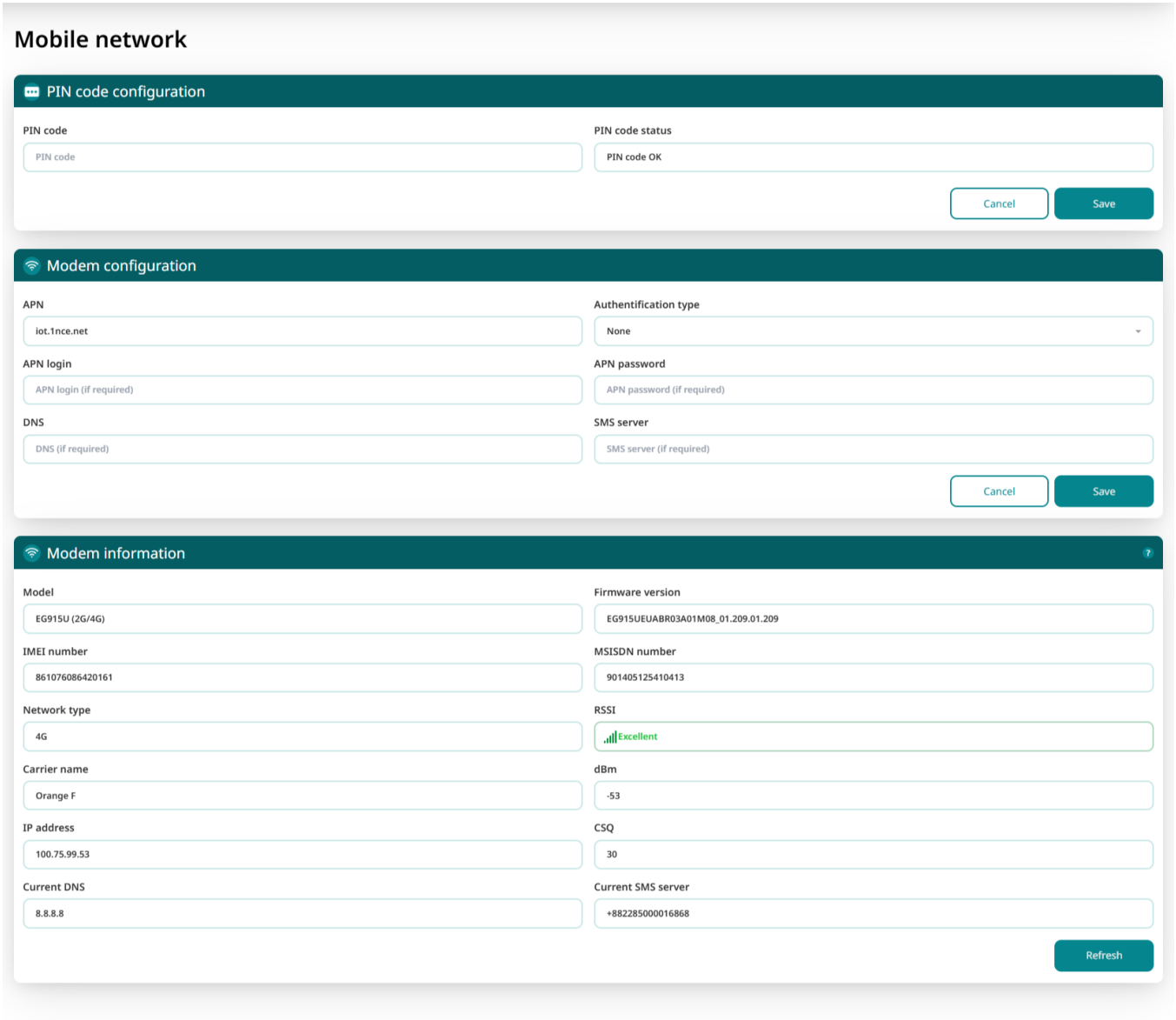
Serial 1 / 2 / 3

- **OK**: All reported devices are connected
- **Partiel**: At least one device is not responding
- **KO**: No devices are responding

- **Non configuré**: No equipment reported

Modem Connection Diagnosis

From the **Network>Mobile** page, you can run a modem communication diagnostic; the various steps in the diagnostic process are:



mobile page

Carte SIM

The different PIN statuses are as follows:

- **Pin Code OK**: The modem can access the SIM card
- **Pin code Required**: The SIM card is waiting for a code
- **Unkown**: Various modem errors
- **PUK code required**: The SIM card is locked
- **SIM card not inserted**: No SIM card inserted

Réception et type de réseau

From the **Modem information** tab, you can also view the signal strength and the type of network the modem is connected to:

- **RSSI**: Indication of the modem's received signal strength, shown as a number of bars
- **Network Type**: The type of network to which the modem is connected (2G/3G/4G)

Connexion DATA et abonnement

When the APN is configured correctly, an IP address is assigned to the modem, and this information is displayed in the same section.

> **Note**
>
> If no IP address is displayed, you will not be able to connect to the servers.

> **Note**
>
> If the SIM card has exceeded its data limit, it will also be impossible to connect to the servers.

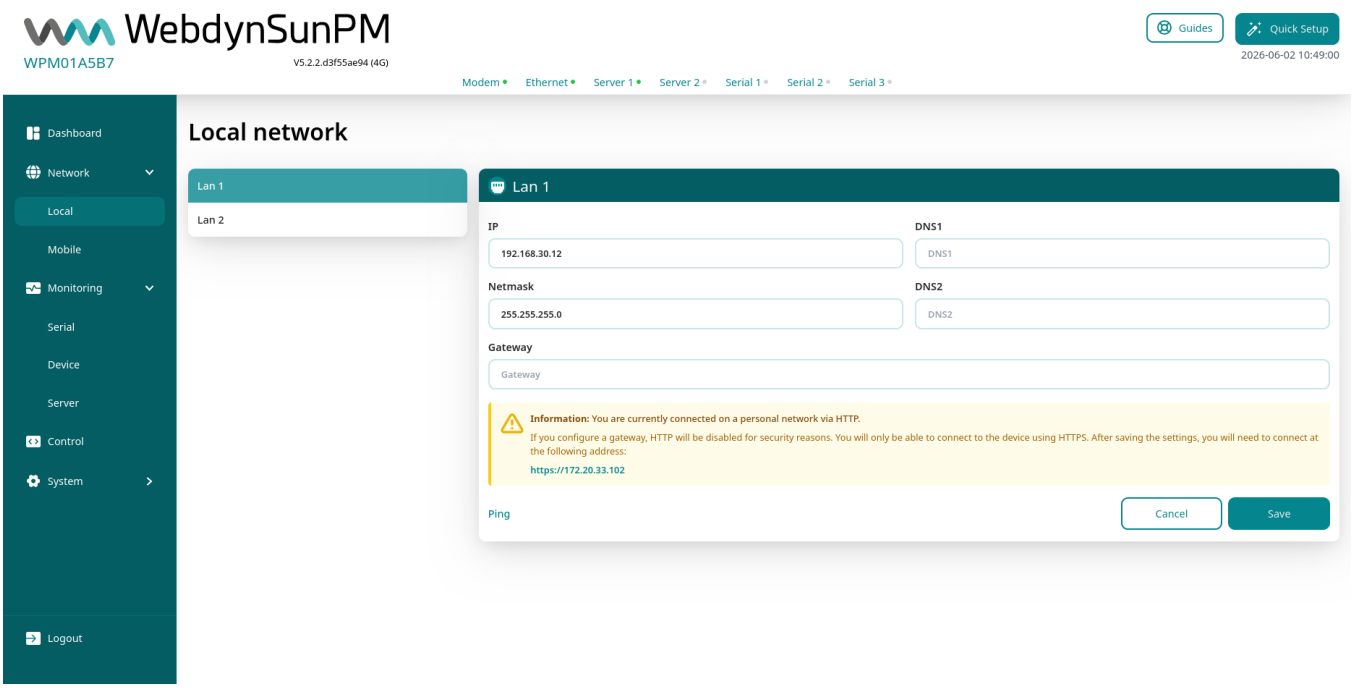
Commande `status` par SMS

You can also use the SMS command `status` to obtain information about the gateway's mobile connectivity:

- **apn**: Name of the APN configured for the modem.
- **ipMobile**: The modem's IP address.
- **csq**: Modem signal strength.

Ethernet Connection Diagnosis

From the **Network>Local** page, you can test connectivity to the gateway/router.



network page

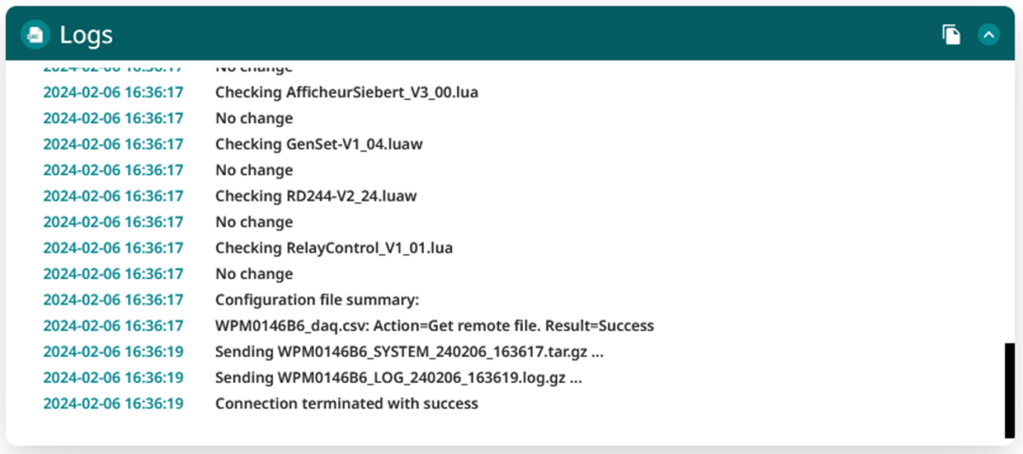
The **Ping** button allows you to test whether the gateway configured on the selected interface is reachable.

If the gateway's IP address has been changed without being recorded, you must perform a **RESET IP** using the gateway's physical button.

6.5 Logs

6.5.1 Log Files

When the connection is initiated via the local web interface, you can monitor the connection process in the **Logs** section.



log server connection

Every time the gateway connects to a server—whether via modem or Ethernet—all operations are logged for future reference.

Below is an example of a connection log:

```
2024-02-11 14:25:03:Firmware version : WebdynSunPM 3.2.11.34874
2024-02-11 14:25:03:Connection 1 wait for exclusive...
2024-02-11 14:25:03:Connection requested
2024-02-11 14:25:16:Ntp synchronisation success
2024-02-11 14:25:16:Connection to 192.168.2.13:test server in progress …
2024-02-11 14:25:16:Alarms management…
2024-02-11 14:25:16:Data acquisition management…
2024-02-11 14:25:16:Sending WPM00BDE4_Modbus_210211_142503.csv.gz
2024-02-11 14:25:16:Data acquisition management…
2024-02-11 14:25:16:Sending WPM00BDE4_IO_210211_142507.csv.gz
2024-02-11 14:25:16:Sending WPM00BDE4_SUNSPEC_210211_142507.log.gz
2024-02-11 14:25:16:Configuration management…
2024-02-11 14:25:16:Checking WPM00BDE4_config.ini
2024-02-11 14:25:16:No change
2024-02-11 14:25:16:Checking WPM00BDE4_daq.csv
2024-02-11 14:25:16:Sending local changes
2024-02-11 14:25:16:Remove file
2024-02-11 14:25:17:Checking WPM00BDE4_var.ini
2024-02-11 14:25:17:No change
2024-02-11 14:25:17:Checking WPM00BDE4_IO.csv
2024-02-11 14:25:17:No change
2024-02-11 14:25:17:Checking WPM00BDE4_SunSpec_inverter_SMA_Solar_Inverter_9301_ModbusTCP.csv
2024-02-11 14:25:17:Getting remote changes
2024-02-11 14:25:21:Failed to parse keep open parameter () for device
192.93.121.23:502, slave: 126, use default value true
```

```
2024-02-11 14:25:21:Local changes sent WPM00BDE4_daq.csv
2024-02-11 14:25:21:Script management...
2024-02-11 14:25:21:Checking WPM00BDE4_scl.ini
2024-02-11 14:25:21:Getting remote changes
2024-02-11 14:25:21:Checking Test script 1.lua
2024-02-11 14:25:21:No change
2024-02-11 14:25:21:Checking Test script 2.lua
2024-02-11 14:25:21:No change
2024-02-11 14:25:21:Configuration file summary:
2024-02-11 14:25:21:WPM00BDE4_SunSpec_inverter_SMA_Solar_Inverter_9301_ModbusTCP.csv: Action=Get
remote file. Result=Success
2024-02-11 14:25:21:WPM00BDE4_daq.csv: Action=Send local file.
Result=Success
2024-02-11 14:25:21:WPM00BDE4_scl.ini: Action=Get remote file.
Result=Success
2024-02-11 14:25:24:Sending WPM00BDE4_SYSTEM_210211_142521.tar.gz
```

System Information and NTP Synchronization

First and foremost, there is information about the gateway's software version, followed by the connection to the NTP server for time synchronization.

```
2024-02-11 14:25:03:Firmware version : WebdynSunPM 3.2.11.34874
2024-02-11 14:25:03:Connection 1 wait for exclusive...
2024-02-11 14:25:03:Connection requested
2024-02-11 14:25:16:Ntp synchronisation success
```

Sending Alarms and Data

`2024-02-11 14:25:16:Alarms management…`

Data upload

```
2024-02-11 14:25:16:Data acquisition management…
2024-02-11 14:25:16:Sending WPM00BDE4_Modbus_210211_142503.csv.gz
2024-02-11 14:25:16:Data acquisition management…
2024-02-11 14:25:16:Sending WPM00BDE4_IO_210211_142507.csv.gz
2024-02-11 14:25:16:Sending WPM00BDE4_SUNSPEC_210211_142507.log.gz
```

Synchronizing Configuration Files

```
2024-02-11 14:25:16:Configuration management…
2024-02-11 14:25:16:Checking WPM00BDE4_config.ini
2024-02-11 14:25:16:No change
2024-02-11 14:25:16:Checking WPM00BDE4_daq.csv
2024-02-11 14:25:16:Sending local changes
2024-02-11 14:25:16:Remove file
2024-02-11 14:25:17:Checking WPM00BDE4_var.ini
2024-02-11 14:25:17:No change
2024-02-11 14:25:17:Checking WPM00BDE4_IO.csv
2024-02-11 14:25:17:No change
2024-02-11 14:25:17:Checking WPM00BDE4_SunSpec_inverter_SMA_Solar_Inverter_9301_ModbusTCP.csv
2024-02-11 14:25:17:Getting remote changes
2024-02-11 14:25:21:Failed to parse keep open parameter () for device
192.93.121.23:502, slave: 126, use default value true
2024-02-11 14:25:21:Local changes sent WPM00BDE4_daq.csv
```

The log file indicates that a number of files were checked and that no changes were detected. It then indicates that the `WPM00BDE4_SunSpec_inverter_SMA_Solar_Inverter_9301_ModbusTCP.csv` file has been modified on the remote server. It is then retrieved, read, and imported locally.

Changes were also detected in the equipment, and the `WPM00BDE4_daq.csv` file is being sent to the server

Script Synchronization

The process concludes with the management of script files:

```
2024-02-11 14:25:21:Script management...
2024-02-11 14:25:21:Checking WPM00BDE4_scl.ini
2024-02-11 14:25:21:Getting remote changes
2024-02-11 14:25:21:Checking Test script 1.lua
2024-02-11 14:25:21:No change
2024-02-11 14:25:21:Checking Test script 2.lua
2024-02-11 14:25:21:No change
```

All scripts are checked on the server. As indicated in the log, there were no changes here.

End of connection

Once all processing is complete, a summary provides an overview of all the operations that took place:

```
2024-02-11 14:25:21:Configuration file summary:
2024-02-11 14:25:21:WPM00BDE4_SunSpec_inverter_SMA_Solar_Inverter_9301_ModbusTCP.csv: Action=Get
remote file. Result=Success
2024-02-11 14:25:21:WPM00BDE4_daq.csv: Action=Send local file.
Result=Success
2024-02-11 14:25:21:WPM00BDE4_scl.ini: Action=Get remote file.
Result=Success
2024-02-11 14:25:24:Sending WPM00BDE4_SYSTEM_210211_142521.tar.gz
```

It states that the following files were transferred from the server:

- `WPM00BDE4_SunSpec_inverter_SMA_Solar_Inverter_9301_ModbusTCP.csv`
- `WPM00BDE4_scl.ini` It also states that the file `WPM00BDE4_daq.csv` was sent to the server.

The log file ends with a note indicating that the system logs have been sent

6.5.2 Equipment Communication Diagnostic Logs

When the diagnostic tool is enabled, a diagnostic log will be sent to the configured servers the next time a connection is made, and will be stored in the logs directory (by default, `/LOG`).

Serial and Ethernet Interface Logs

The names of the log files for communications with devices are structured according to the same principle as the gateway's other log files, namely: `<UID>_interface_date.log.gz`.

The interface is the same as the one chosen for the logs, namely:

- Serial1
- Serial2
- Serial3

- Ethernet

The logs contain the following information:

```
datetime_1;data
datetime_2;data
...
datetime_Y;data
```

With:

- **datetime**: timestamp in **DD/MM/YYYY hh:mm:ss.mmm** format
- **data**: the transmitted data and its direction
  - **serial** interface: data in hexadecimal
  - **ethernet** interface: device IP address and data in hexadecimal
  - `<=` : Outgoing communication
  - `=>` : Incoming communication

> **Note**
>
> It should be noted that, in the case of Modbus, the complete Modbus TCP frame is provided if the connection is an Ethernet connection. Otherwise, it is the Modbus RTU frame.

In the case of Modbus errors, the frame may be prefixed with the following messages:

- `*** CRC ***` : A CRC error was detected in the incoming frame. The frame is therefore invalid. This is a hardware communication error. Too many CRC errors indicate a problem with the installation. The frame is ignored.
- `*** BAD SLAVE ***` : A slave with an incorrect number responded to the request. This may be due to a configuration error in a device, which could disrupt the proper operation of the system. The frame is ignored.
- `*** EXCEPTION ***` : The slave returned an exception in response to the request. This means that the request that was sent is invalid for the device in question.
- `*** INVALID ID ***` : A slave responded to a ModbusTCP request with an incorrect identifier. The frame was ignored.
- `*** INVALID FCT ***` : The response contains an invalid function code. The frame is ignored.

Log of Entries and Exits:

The names of the input/output log files are structured according to the same principle as the gateway's other log files, namely: `<UID>_IO_date.log.gz` .

The logs contain the following information:

```
datetime_1;data
datetime_2;data
...
datetime_Y;data
```

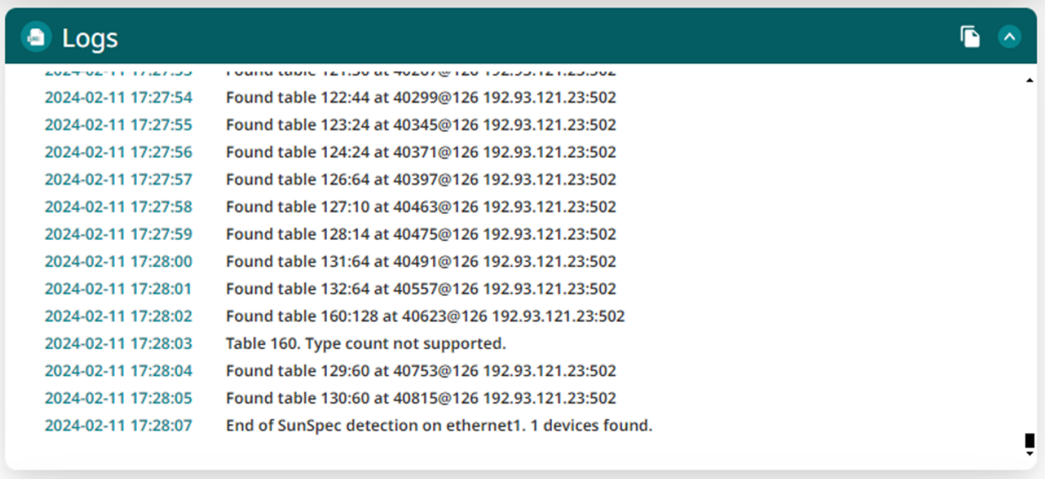
With:

- **datetime**: timestamp in **DD/MM/YYYY hh:mm:ss.mmm** format
- **data**: Data regarding inputs and outputs:
  - **In**: indicates that this is an entry
  - **Out**: indicates that this is an output

- **name**: the name of the input/output as configured in the definition file
- **0**: The input/output is closed
- **1**: The input/output is open
- **valeur**: the value of an analog input

6.5.3 SunSpec Detection Logs

During a SunSpec detection, the detection results are displayed in real time on the local web:



sunspec detection

This log may also be found in the logs directory (by default, `/LOG`) the next time you connect to the server. The file name is `<UID>_SUNSPEC_timestamp.log.gz`

The log file contains exactly the same information as what was displayed on the screen:

```
24/02/11-17:27:28;SunSpec detection start on ethernet1. Estimated time : 621 seconds.
24/02/11-17:27:30;NON SUNSPEC device found at @1 192.93.121.23:502.
24/02/11-17:27:31;NON SUNSPEC device found at @2 192.93.121.23:502.
24/02/11-17:27:32;NON SUNSPEC device found at @3 192.93.121.23:502.
24/02/11-17:27:47;SunSpec device found at @126 192.93.121.23:502.
24/02/11-17:27:48;Found table 1:66 at 40004@126 192.93.121.23:502
24/02/11-17:27:49;Found table 11:13 at 40072@126 192.93.121.23:502
24/02/11-17:27:50;Found table 12:98 at 40087@126 192.93.121.23:502
24/02/11-17:27:51;Found table 101:50 at 40187@126 192.93.121.23:502
24/02/11-17:27:52;Found table 120:26 at 40239@126 192.93.121.23:502
24/02/11-17:27:53;Found table 121:30 at 40267@126 192.93.121.23:502
24/02/11-17:27:54;Found table 122:44 at 40299@126 192.93.121.23:502
24/02/11-17:27:55;Found table 123:24 at 40345@126 192.93.121.23:502
24/02/11-17:27:56;Found table 124:24 at 40371@126 192.93.121.23:502
24/02/11-17:27:57;Found table 126:64 at 40397@126 192.93.121.23:502
24/02/11-17:27:58;Found table 127:10 at 40463@126 192.93.121.23:502
24/02/11-17:27:59;Found table 128:14 at 40475@126 192.93.121.23:502
24/02/11-17:28:00;Found table 131:64 at 40491@126 192.93.121.23:502
24/02/11-17:28:01;Found table 132:64 at 40557@126 192.93.121.23:502
24/02/11-17:28:02;Found table 160:128 at 40623@126 192.93.121.23:502
24/02/11-17:28:03;Table 160. Type count not supported.
24/02/11-17:28:04;Found table 129:60 at 40753@126 192.93.121.23:502
24/02/11-17:28:05;Found table 130:60 at 40815@126 192.93.121.23:502
24/02/11-17:28:07;End of SunSpec detection on ethernet1. 1 devices found.
```

It should be noted that the SunSpec detection log files are in "CSV" format so they can be easily imported into a spreadsheet.

> **Note**
>
> The SunSpec detection log is in the `csv` format

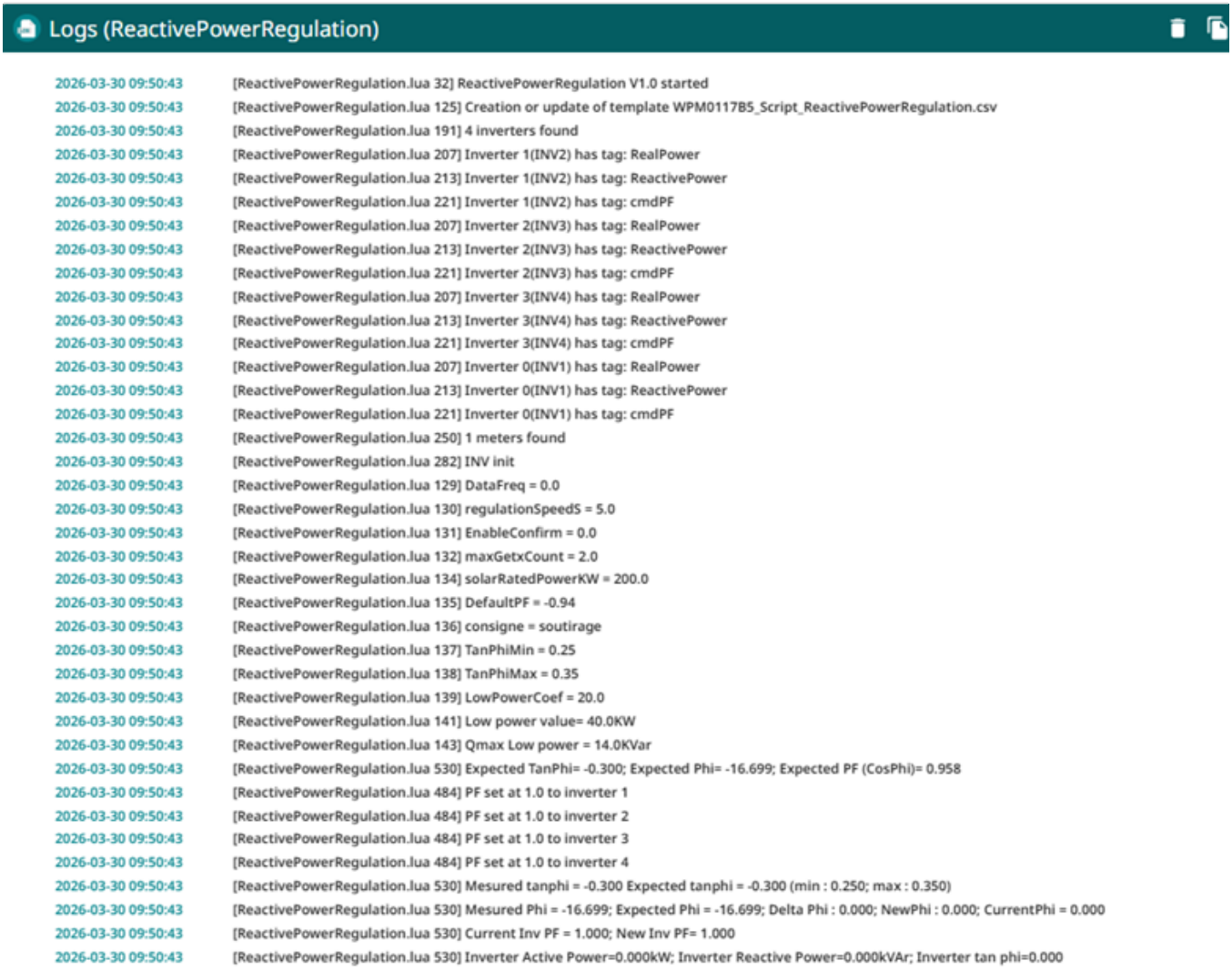
6.5.4 Script Logs

Each script runs in a separate environment. As a result, each one has a log file that is automatically generated when it runs.

This log file can be viewed in two ways:

- Either directly on the web server on the **Control** page by clicking the **Script logs** menu for additional script options

script logs



script logs

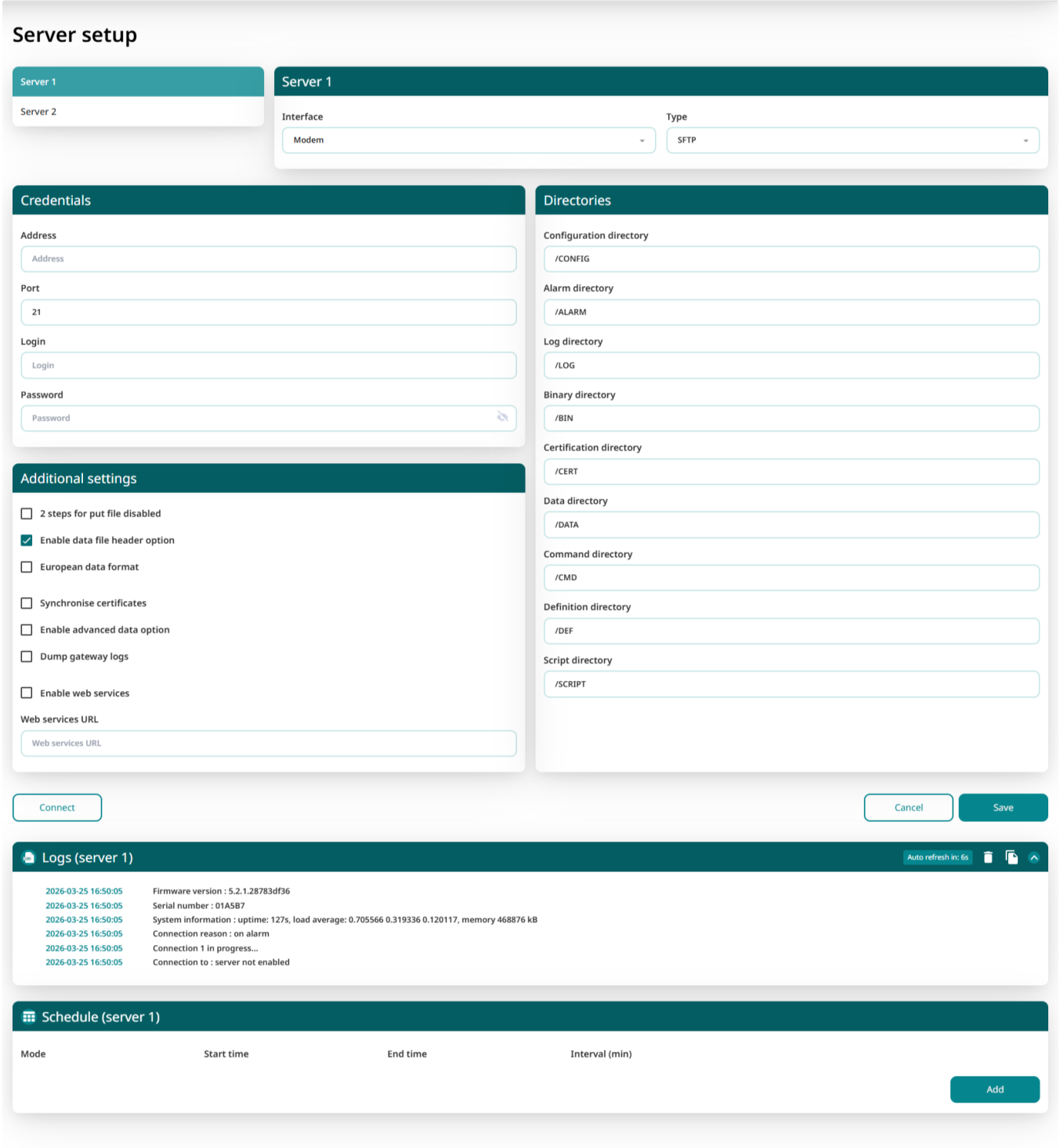
- Either on the server, in the log files. The logs shown above are also available on the server under the name `<UID>_LUA_<script_name>_timestamp.log.gz`

> **Note**
>
> Please note that the script log files are in the `csv` format.

6.5.5 System Logs

System logs are the gateway's internal operating logs. They contain internal debugging information that may be requested by Webdyn support.

By default, system logs are not transferred to the server. To enable this, you must activate the **Dump gateway logs** option on the server configuration page:



server page

System log files are also automatically transferred when a connection is forced through this same page by clicking the **Connect** button or via the **connect** SMS command.

The file name is formatted as `<UID>_SYSTEM_timestamp.log.gz`

7 Annexes

7.1 Configuration Files

7.1.1 <uid>_config.ini

This file is used to configure the servers and the gateway:

- Access to Servers
- Server Configuration
- Server File System Directory Structure
- NTP Clock Synchronization
- gateway Information
- Update File

This file is automatically created during configuration with the embedded web server.

> **Note**
>
> You do not need to fill in all the fields

Update File

- **BIN_FileName**: the name of the firmware update file. This parameter cannot be empty if **BIN_Checksum** is specified. The firmware must then be located in the corresponding directory on the server.
- **BIN_Checksum**: Specifies the checksum of the update file. This parameter cannot be empty if **BIN_FileName** is specified.

gateway Information

- **Concentrator_Identifier**: gateway ID. If this field is left blank, the default ID is used.
  - Default value: `WPMxxxxxx` (xxxxx represents the last characters of the serial number)
- **WEB_Password**: Password associated with the `userhigh` login on the embedded web server
- **SMS_Password**: Password used to encrypt SMS messages received by the gateway
- **LOG_Level**: Logging level recorded in the system files.
  - Default value: `3`

NTP Clock Synchronization

Synchronization with a remote NTP server is necessary to accurately time-stamp the collected data.

- **NTP_Server1**: Primary clock synchronization server
  - Default value: `pool.ntp.org`
- **NTP_Server2**: Secondary clock synchronization server
- **NTP_TimeZone**: Time zone for timestamping in conjunction with the NTP server
  - Default value: `UTC`

Server File System Directory Structure

A directory tree must be created in advance for servers 1 and 2 configured for SFTP, FTP, or HTTPS WebDAV

The parameter format is:

```
SERVER_PARAMETER = VALUE
```

With `SERVER = { FTP, FTP2, HTTP, HTTP2 }`

This is how `FTP_DirConfig` configures the configuration directory for FTP Server 1.

The list of parameters is as follows

- **DirConfig**: Configuration Files Directory
  - Default value: `/CONFIG`
- **DirAlarm**: Alarm Files Directory
  - Default value: `/ALARM`
- **DirLog**: : Log file directory
  - Default value: `/LOG`
- **DirBin**: Directory containing firmware update files
  - Default value: `/BIN`
- **DirData**: Data Files Directory
  - Default value: `/DATA`
- **DirScript**: Script Files Directory
  - Default value: `/SCRIPT`
- **DirCmd**: Directory of command files
  - Default value: `/CMD`
- **DirDef**: Directory containing definition files
  - Default value: `/DEF`
- **DirCert**: Certificate file directory
  - Default value: `/CERT`

Server Configuration

Serveur 1 et 2

Two servers can be configured on the gateway.

The parameter format:

```
SERVERSLOT_PARAMETER = VALUE
```

With `SERVERSLOT = { SERVER, SERVER2 }`

Thus, `SERVER_Address` configures the IP address (or URL) of Server 1.

The list of parameters is as follows:

- **Address**: URL (if DNS is configured) or the server's IP address.
- **Interface**: Network interface used. Possible values:
  - **modem**
  - **ethernet**
- **Type**: Server types. Possible values:
  - **ftp**
  - **sftp**
  - **webdav**
  - **mqtt** (server 2 only)
  - **mqtts** (server 2 only)
  - **mqtts_azure** (server 2 only)
  - **mqtts_aws** (server 2 only)
  - **mqtts_gcloud** (server 2 only) (obsolete)

Serveur SFTP/FTP/HTTPS WebDAV

The parameter format is:

```
SERVER_PARAMETER = VALUE
```

With `SERVER = { FTP, FTP2, HTTP, HTTP2 }`

This is how `FTP2_Password` sets the password for FTP server 2

The list of parameters is as follows:

- **Login**: The username to use to log in to the server. **Obligatoire**
- **Password**: The password associated with the login. **Obligatoire**
- **Port**: Port to use to connect to the server. **Obligatoire**
- **EnableAdvancedData**: Adds the number of readings taken during acquisition to the data files. Possible values:
  - **0** (default): disabled
  - **1**: enabled
- **TwoStepsSendingDisabled** (SFTP/FTP only): Two-step upload using a temporary file until the file is complete. Possible values:
  - **0** (default): enabled
  - **1**: disabled
- **HeaderOption**: Add headers to data files. Possible values:
  - **0** (default): no header
  - **1**: with header
- **EuroDateFormat**: Timestamp format. Possible values:
  - **0** (default): ISO format (YY/MM/DD-HH:MM:SS)
  - **1**: European format (DD/MM/YY-HH:MM:SS)
- **SynchroniseCertificates**: Certificate synchronization. Possible values:
  - **0** (default): Synchronization disabled
  - **1**: Synchronization enabled
- **UploadLog**: Sending log files. Possible values:
  - **0 (défaut)**: disabled
  - **1**: enabled
- **WebServicesEnable**: Enable web services (FTP/SFTP only). Possible values:
  - **0 (défaut)**: disabled
  - **1**: enabled
- **WebServicesUrl**: Web service return address (FTP/SFTP only). The URL must be in the format `http://adresse/page`

Serveur MQTT(s)

> **Warning**
>
> MQTT(s) is available only on server 2

The parameter format is:

```
SERVER_PARAMETER = VALUE
```

With `SERVER = { MQTT2 }`

Thus, `MQTT2_Login` configures the username for MQTT Server 2.

- **Login**: Username for accessing the MQTT server. Only for server types **mqtt** and **mqtts**
- **Password**: Password to access the MQTT server. Only for server types **mqtt** and **mqtts**

- **Port**: MQTT server port
- **Topic**: Topic name for data submitted by the concentrator. All types except **mqtts_azure**
- **ResultTopic**: Topic name for the results of commands sent to the gateway. The *ControlTopic* parameter must be specified to use commands. If specified, the MQTT connection is persistent. All types except **mqtts_azure**
- **QoS**: Quality of Service—guarantees that messages are received correctly.
  - **0**: The message will be sent only once, with no guarantee of delivery.
  - **1**: The message will be sent until the broker confirms receipt.
  - **2**: The message will be sent only once, with guaranteed delivery. For types **mqtts_azure** and **mqtts_aws**, QoS 2 is not supported.
- **Timeout**: Maximum timeout in seconds for a response from the MQTT server. If this time is exceeded, the transmission is stopped and will be retried during the next connection. Applies only to **QoS 1** and **QoS 2**. Default value: 30s
- **TlsVersion**: Version supported by the MQTT server. Possible values are
  - **tlsv1.1**: TLS v1.1
  - **tlsv1.2** (default): TLS v1.2
- **AlarmTopic**: Topic name for publishing alarms. If specified, the MQTT connection is persistent. Any type except **mqtts_azure**
- **CaCertFile**: Name of the certificate used for authentication with the MQTT server. The certificate must be imported into the gateway via SFTP, FTP, WebDAV, HTTPS, or the web interface. Any type except **mqtt**
- **CertFile**: The certificate name specific to the gateway. The certificate must be imported onto the gateway via SFTP, FTP, WebDAV, HTTPS, or the web interface. Any type except **mqtt**.
- **CloudDevice**: A unique, customizable identifier for the device in a registry defined on the MQTT server.
  - Corresponds to `deviceId` on Google IoT Cloud (deprecated).
  - Corresponds to `device_id` on Azure IoT gateway. Only for type **mqtts_azure**.
- **CloudProjectId**: A unique, customizable project identifier defined on the MQTT server. For **mqtts_azure** types only
  - The name of **Iot gateway** on Azure IoT gateway.
- **ClientId**: Customizable device identifier on the MQTT server. Only for types **mqtt** and **mqtts**
- **ControlTopic**: Name of the topic for commands to be retrieved by the gateway. The MQTT2_ResultTopic parameter must be set to use commands. If set, a persistent connection is established. Works with all MQTT types except **mqtts_azure**.
- **EnableAdvancedData**: Publishes the number of complete reads for this acquisition period in the data topic. Possible values are:
  - **0** (default): Disabled
  - **1**: Enabled

- **EnableAlarmPost**: Enable alarm publishing to the dedicated topic. Only for type **mqtts_azure**. Possible values are:
  - **0**: Disabled
  - **1**: Enabled (permanent connection)
- EnableInvokeMethod**: Enable method calls. For **mqtts_azure** types only. Allows the use of dedicated topics. Possible values are:
  - **0**: Disabled
  - **1**: Enabled (permanent connection)
- **Insecure**: Disables verification of the hostname specified in certificates. Only for type **mqtts**. Possible values are:
  - **0**: Verification enabled
  - **1**: Verification disabled.
- **KeepAlive**: Idle timeout (in seconds) between the client and the broker. If the timeout is exceeded, the client sends a ping to check the connection. If the value is set to "0," KeepAlive is disabled. If the gateway is in persistent connection mode with the MQTT server and a disconnection is detected after a KeepAlive, the gateway will automatically reconnect to the MQTT server.
- **KeyFile**: Name of the file containing the private key specific to the gateway used for the connection. The file can be retrieved from your MQTT server and must be imported to the gateway via FTP or the web interface. All types except **mqtt**

7.1.2 <uid>_daq.csv

The **<uid>_daq.csv** file contains the settings for all interfaces and the list of devices configured on the gateway:

- Modem
- Ethernet
- Serial bus
- Modbus TCP slave
- Connected devices

After the initial setup with the built-in web server, the file will be created the first time you connect to the remote server.

If a file already exists on the remote server, the local configuration file will be deleted.

> **Info**
>
> The format of this file is CSV (comma-separated values), which uses the semicolon `;` as a delimiter.

> **Warning**
>
> Any other CSV delimiter—such as the comma `,`—will be rejected by the gateway.

Modem Configuration

The modem configuration is based on the following settings:

- **type**: Interface type. The only possible value is `MODEM`

- **pin**: Modem PIN code, if set
- **apn**: The APN address of the mobile operator that enables the IP connection. This parameter is required.
- **login**: Login provided by the mobile operator. This field may be empty, depending on the operator
- **password**: Password provided by the mobile operator. This field may be empty, depending on the operator
- **authentication**: Authentication method provided by the mobile operator. Possible values are:
  - **none**: No authentication required
  - **PAP**: Requires a username and password
  - **CHAP**: Requires a username and password
  - **Both**: Requires a username and password
- **server**: SMS message center, if different from the one provided by the mobile carrier. This field may be left blank.
- **dns**: IP address of the DNS server used to resolve domain names if different from the one provided by the mobile carrier (e.g., Google DNS 8.8.8.8). This field may be left blank

Below is a standard example of a modem configuration:

```
type;pin;apn;login;password;authentication;server;dns
MODEM;;m2minternet;;;none;;
```

Ethernet Configuration

Configuring Ethernet connections requires the following settings:

- **type**: Interface type. Possible values are:
  - **LAN1**: Corresponds to the LAN1 Ethernet port
  - **LAN2**: Corresponds to the LAN2 Ethernet port
- **ip**: Local IP address. Default value:
  - **LAN1**: 192.168.1.12
  - **LAN2**: 192.168.2.12
- **mask**: Subnet mask. Default value:
  - **LAN1**: 255.255.255.0
  - **LAN2**: 255.255.255.0
- **gateway**: Address of the gateway used to communicate with devices outside the configured local network. This field may be left blank.
- **DNS1**: IP address of the primary DNS server used to resolve domain names
- **DNS2**: IP address of the secondary DNS server used to resolve domain names

The following is an example of an Ethernet configuration:

```
type;ip;mask;gateway;dns1;dns2
LAN1;192.93.121.37;255.255.255.0;192.93.121.1;192.93.121.8;
LAN2;192.168.2.12;255.255.255.0;;;
```

In this example, the first network interface (LAN 1) is configured with the IP address **192.93.121.37**, with a subnet mask of **255.255.255.0**, allowing it to access all machines connected using the address **the 193.93.121.xyz**.

This interface also uses a router at the address **192.93.121.1** to communicate with external devices and a DNS server accessible at the address **193.93.121.8**. DNS2 is not configured.

The second network interface is left with its default configuration.

Modbus TCP Interframe

If devices are connected via Modbus TCP, it may be necessary to add a delay between requests to prevent overloading the device being read. The **tcpinterFrameMs** parameter allows you to do this.

- **tcpinterFrameMs**: delay in ms between two Modbus TCP requests. Default value: 0

```
tcpInterFrameMs;<value>
```

> **Note**
>
> This setting applies globally to all Modbus TCP communication on both interfaces; if the timeout is too long, it may cause communication to slow down.

Serial Configuration

Configuring serial buses requires the following settings:

- **type**: The interface type. Possible values are:
  - **SERIAL1**: SERIAL1 bus configuration
  - **SERIAL2**: SERIAL2 bus configuration
  - **SERIAL3**: SERIAL3 bus configuration
- **baudrate**: The baud rate. Possible values are: 1200, 2400, 4800, 9600, **19200 (défaut)**, 38400, 57600, 115200, 230400, 460800.
- **data_bits**: Number of data bits per byte. Possible values are: 7,**8 (défaut)**
- **parity**: Parity to be applied to validate the transmitted data. Possible values are:
  - **N**: No (default)
  - **O**: odd
  - **E**: even
- **stop_bits**: Number of stop bits between two bytes. Possible values are: 1 (default), 2.
- **wires**: The number of wires used for communication. Possible values are:
  - **2 (défaut)**: One pair is used for transmission and reception
  - **4**: Two pairs are used, one for transmission and the other for reception
- **protocol**: The protocol used on the link. Possible values are:
  - **Modbus**
  - **proprietary protocol**: See the corresponding appendices
- **interFrame(ms)**: Time interval between two requests. Default value: 0
- **forwarded_port**: Tunnel between Ethernet and serial on the configured port. Modbus TCP requests sent on this port are relayed as Modbus RTU, and Modbus RTU responses are then relayed as Modbus TCP on the same port.

> **Note**
>
> The **forwarded_port** setting is only accessible via a local network; a modem connection cannot be used.

> **Note**
>
> When the gateway receives a response from the device, it imposes a delay equal to the **InterFrame** parameter between the last byte of the last received frame and the first byte of the next frame it sends to that device. This delay applies to the entire bus. Thus, if a new frame is sent to a different device than the previous one, the delay will still be applied.

An example is given below:

```
type;baudrate;data_bits;parity;stop_bits;wires;protocol;interframe(ms);forwarded_port
SERIAL1;9600;8;N;1;2;Modbus;0;
SERIAL2;1200;8;N;1;4;SMANET;0;
SERIAL3;19200;8;N;1;2;PW1;0;
```

- The SERIAL1 connection is configured for 2-wire Modbus RTU at a baud rate of 9600 baud
- The SERIAL2 connection is configured to use the proprietary 4-wire SMANET protocol at a speed of 1,200 baud.
- The SERIAL3 connection is configured to use the proprietary PW1 (PowerOne) protocol at a baud rate of 19,200 baud.

Modbus TCP Slave Configuration

Configuring the Modbus TCP slave requires the following settings:

- **enabled**: Enables the Modbus TCP slave. Possible values are:
  - **0 (défaut)**: disabled
  - **1**: enabled
- **port**: Port in use. Default value: **502**
- **mapping file**: Name of the file containing the definitions of registers and variables. The file must be located in the `/DEF` folder; if this folder does not exist, only the Webdyn registers and variables will be accessible.

Here is an example:

```
ModbusSlave;<enabled>;<port>;<mapping file>
```

Equipment Configuration

Configuring connected devices requires the following settings:

- **index**: Unique numerical index of the device. Special case: **IO** corresponds to the gateway's inputs/outputs
- **interface**: interface used by the device. Possible values are:
  - **SERIAL1**: Modbus RTU device on SERIAL1
  - **SERIAL2**: Modbus RTU device on SERIAL2
  - **SERIAL3**: Modbus RTU device on SERIAL3
  - **IP address**: Modbus TCP device configured with the specified IP address
  - **IP address-port**: Modbus TCP device configured with the specified IP address and port
- **name**: Unique device name. Used for the web interface, MQTT communication, and scripts.
- **address**: The device address in Modbus RTU (UnitID). Possible values range from 1 to 254.
- **acqPeriod(s)**: The acquisition time in seconds for equipment data. If the entered value is 0, no data is stored. Default value: 600
- **timeout(ms)**: The timeout in milliseconds before the request fails. Default value: 1000
- **serialNumber**: Equipment serial number. Filled in automatically upon detection.
- **parameters**: For proprietary protocols, see the appendices. For Modbus TCP devices, the possible values are:
  - **0**: The connection is closed after each request
  - **1**: The connection is maintained after the request to save time on the next request
- **category(RO)**: category associated with the equipment
  - **tag (V5.1.x)**: label associated with the equipment
  - **category** (prior to v5.1.0): category associated with the equipment
- **node**: Node associated with the equipment
  - **model** (prior to v5.2.x): Equipment model. This field is automatically populated by the gateway

- **defFile**: Name of the file containing the definitions of the equipment's registers and variables.

Cas particulier IO

The gateway's internal inputs and outputs are also declared in this section, but only the parameters are specified:

- **index**: IO - not editable
- **name**: Io - not editable
- **acqPeriod(s)**: editable
- **tag**: WebdynSunPM - cannot be modified
- **model**: ioSunPM
- **defFile**: <uid>_IO.csv

The other fields are left blank.

Here is an example:

```
index;interface;name;address;acqPeriod(s);timeout(ms);serialNumber;parameters;category(RO);node;
defFile
IO;;io;;600;;;;ioSunPM;NONE;WPM01A5B7_IO.csv

1;192.168.30.1:1502;Inverter 1;1;600;1000;;1;Inverter;SOLAR;WPM01A5B7_Huawei_V4.csv
2;192.168.30.2:1502;Inverter 2;1;600;1000;;1;Inverter;SOLAR;WPM01A5B7_Huawei_V4.csv
```

Three devices are configured:

- **IO**: Corresponds to the gateway's internal inputs/outputs
  - **acqPeriod(s)**: 600 - recording every 10 hours
  - **defFile**: WPM01A5B7_IO.csv - the name of the definition file describing the gateway's internal input/output configuration
- **1**: Device communicating over Ethernet using Modbus TCP
  - **interface**: 192.168.30.1:1502 - Ethernet device using the default IP address and port
  - **name**: Inverter 1 - the name displayed on web pages
  - **address**: 1 - This device responds to Modbus address 126
  - **acqPeriod(s)**: 600 - data logging every 10 minutes
  - **timeout(ms)**: 1000 - 5-second timeout before the request fails
  - **serialNumber**: empty - not applicable
  - **parameters**: 1 - connection maintained
  - **category(RO)**: Inverter
  - **node**: SOLAR
  - **defFile**: WPM01A5B7_Huawei_V4.csv

7.1.3 <UID>_license.ini

The `<UID>_licence.ini` file contains the licenses for the Webdyn Lua scripts `.luaw`.

The license file must be placed in the configuration directory (default `/CONFIG`) on the remote server. The license file allows you to obtain specific scripts designed by Webdyn. When connecting to the remote server, if the file is detected, it is downloaded and the license is applied immediately. WebdynSunPM will not re-download the license file if it has been deleted.

In that case, please contact Webdyn's sales department, which will be able to advise you and direct you to the appropriate contacts: contact@webdyn.com

Note

> The license file is specific to a single WebdynSunPM. It is not possible to use the same file on multiple gateways. Modifying the contents of the license file is prohibited, as doing so may prevent the gateway from managing licenses.

7.1.4 <UID>_scl.ini

The `<UID>_scl.ini` file contains a list of the scripts configured on the gateway.

If the basic configuration is performed with the embedded web server, this file is automatically created when the gateway connects for the first time.

When connecting to the remote server, if the file is detected, it is downloaded and the configuration is applied immediately, regardless of the local configuration.

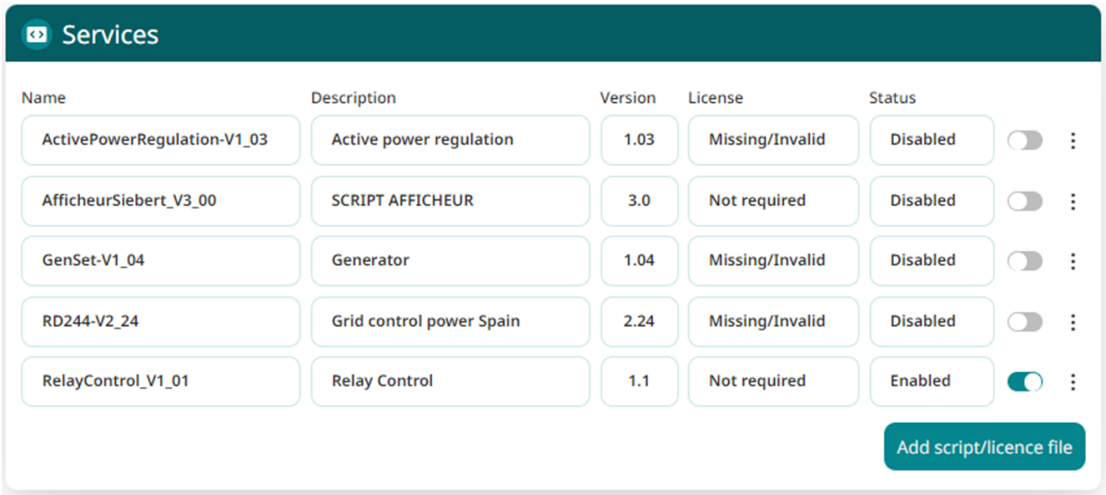
The file contains three lines for each configured script. Each line contains the script number.

The format is as follows:

Variable	Definition	Default value
`SCRIPT_File[n]`	Scenario Name	Script file name with the extension
`SCRIPT_Enable[n]`	Script Activation	0
`SCRIPT_Args[n]`	Script parameter	

Where `n` is replaced by the script's index number. This number starts at 0.

So, with the following script configuration:



services

```
SCRIPT_Args[0]={"solarRatedPowerKW":"110","gridRegulationType":"consumption","gridRegulationTargetKW":"50","gridEffectiveRegulationPercent":"20","regulationSpeedS":5,"phaseControl":"sum","errorAction":"none"}
SCRIPT_Enable[0]=0
SCRIPT_File[0]=ActivePowerRegulation-V1_03.luaw
SCRIPT_Args[1]=
SCRIPT_Enable[1]=0
SCRIPT_File[1]=AfficheurSiebert_V3_00.lua
SCRIPT_Args[2]=
SCRIPT_Enable[2]=0
```

```
SCRIPT_File[2]=ControlPower.lua
SCRIPT_Args[3]={"solarRatedPowerKW":"230","maxSolarPowerPercent":"80","startTempoS":"2","regulat
ionSpeedS":"5","dgSafetyKW":"30","gridTargetKW":"60","gridSafetyKW":"20","errorAction":"stop","d
g":[{"dgEnabled":true,"dgRatedPowerkW":"30","dgTargetPercent":"100"},{"dgRatedPowerkW":"30","dgT
argetPercent":"100"},{"dgRatedPowerkW":"30","dgTargetPercent":"100"}]}
SCRIPT_Enable[3]=0
SCRIPT_File[3]=GenSet-V1_04.luaw
SCRIPT_Args[4]=
SCRIPT_Enable[4]=0
SCRIPT_File[4]=RD244-V2_24.luaw
SCRIPT_Args[5]=""
SCRIPT_Enable[5]=1
SCRIPT_File[5]=RelayControl_V1_01.lua
```


Warning

If you manually modify this file, make sure there are no duplicate index numbers; otherwise, the configuration will be rejected.

7.1.5 <uid>_var.ini

The **<uid>_var.ini** file contains a list of the server connection schedules configured on the gateway.

If the basic configuration is performed with the embedded web server, this file is automatically created when the gateway connects for the first time.

When connecting to the remote server, if the file is detected, it is downloaded and the configuration is applied immediately, regardless of the local configuration.

The file contains one line for each configured schedule. Each line contains the schedule number and the corresponding settings.

The format for connecting to Server 1 is as follows:

`SCHEDULE_Params[index]=Id|Type|StartTime|Interval|Count`

Le format pour la connexion au serveur 2 est la suivante:

`SCHEDULE2_Params[index]=Id|Type|StartTime|Interval|Count`

The parameters are:

- **index**: Unique scheduling index (per server). Starts at 0.
- **Id**: Unique identifier (per server) for the configuration line. Starts at 1.
- **Mode**: Recurrence. Possible values:
  - **everyday** (default): log in every day
  - **monday**: Log in every Monday
  - **tuesday**: Log in every Tuesday
  - **wednesday**: Log in every Wednesday
  - **thursday**: Log in every Thursday
  - **friday**: Log in every Friday
  - **saturday**: Log in every Saturday
  - **sunday**: Log in every Sunday
  - **first**: Log in on the 1st of every month
  - **middle**: Log in on the 15th of every month
  - **last**: Log in on the last day of each month
- **StartTime**: Task start time in the `HH:MM:SS` format.
  - Default value: 00:00:00

- **Interval**: Server connection interval in seconds.
  - Default value: 1440
- **Count**: Maximum number of daily connections.
  - Default value: 1

Exemple de fichier <uid>_var.ini

```
SCHEDULE_Params[0]=1|everyday|00:00:00|1440|1
SCHEDULE_Params[1]=2|monday|00:00:00|1440|1
SCHEDULE_Params[2]=3|first|00:00:00|1440|1
SCHEDULE_Params[3]=4|middle|01:00:00|600|2
SCHEDULE_Params[4]=5|last|02:02:00|120|12
SCHEDULE2_Params[0]=1|everyday|00:00:00|1440|1
SCHEDULE2_Params[1]=2|wednesday|12:00:00|1440|1
SCHEDULE2_Params[2]=3|sunday|21:00:00|1440|1
```

7.2 External Commands

7.2.1 Order Reference

SYSTEM

`connect`

SMS/Modbus Slave Command Format

```
connect=<connexion>
```

Settings

Last Name	Required	Description
server	No	Server ID (`1 = Server 1` , `2 = Server 2`)

SMS/Modbus Slave Command Example

```
connect=2
```

SMS/Modbus Slave Format Response

- Success: no return
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.connect",
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```


> **Note**
>
> The `connect` command is not useful for FTP/SFTP/WebDAV servers

`status`

SMS/Modbus Slave Command Format

```
status
```

Settings

- **number**: Phone number that will receive the status SMS sent by the gateway. **only for `json` command file**

SMS/Modbus Slave Format Response

- Success:
  - `type` : Product Name
  - `idSite` : Product ID
  - `FW` : Firmware version
  - `ModeServ1` : Connection mode for Server 1
  - `apn` : APN name
  - `ipMobile` : Mobile IP address
  - `csq` : Modem signal strength
  - `network` : Modem network type
  - `ip` : Ethernet 1 IP addresses
  - `mask` : Ethernet subnet mask 1
  - `gw` : Ethernet Gateway 1
  - `dns` : List of Ethernet 1 DNS servers
  - `srv1` : Server 1 information delimiter
  - `type` : Server 1 connection type (ftp/sftp/webdav/mqtt/mqtts)
  - `addr` : Server 1 address format `login@server-address:server-port`
  - `last` : Server 1—last connection attempt `DD/MM HH:MM < result >`

SMS/Modbus Slave Response Example

- success: next message


```
type=WebdynSunPM 4G
idSite=WPM01A5B7
FW=5.2.1.a05eac38d
ModeServ1=ethernet
apn=iot.1nce.net
ipMobile=100.75.99.53
csq=28,99
network=4G
ip=192.168.30.12
mask=255.255.255.0
```

```
gw=
dns=
srv1:
type=sftp
addr=webdyn@172.20.33.2:2022
last=27/03 17:00 success
```

- Error: error message

`json` command file format

```
{
        "rpcName": "sunpm.status",
        "parameters": {
                "number": "+33700000000"
        },
        "callerId": "1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`factory`

SMS/Modbus Slave Command Format

```
factory
```

Settings None

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.factory",
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`reboot`

SMS/Modbus Slave Command Format

```
reboot
```

Settings None

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.reboot",
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`updateFirmware`

SMS/Modbus Slave Command Format

```
updatefirmware=<url[:port]>:<login>:<password>:<checksum>:<interface>
```

Settings

Last Name	Required	Description
url	Yes	Full URL, including the protocol. **Port is optional and is included in url** (`https://server[:port]`)
login	Yes	Connecting to the server
password	Yes	Server password
checksum	Yes	Firmware Checksum
interface	Yes	`Ethernet` or `Modem`.

SMS/Modbus Slave Command Example

```
updatefirmware=ftp://ftp.url.com:2222/BIN/
wgapp_4.1.0.37427.spm:identifiant:id:70a0eeeae295a7e16d3811b66bee9b66:modem
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.updateFirmware",
        "parameters":{
                "url":"https://www.webdyn.com/download/wgapp_new_fw.spm",
                "login":"identifiant",
                "password":"webdyn",
                "checksum":"78a37fa7f6714876be7d08d0c39a067b",
                "interface":"modem"
        },
        "callerId":"674"
}
```

`json` Response Format

```
{
        "callerId":"674",
        "result":"Firmware downloaded successfully. System will restart..."
}
```

> **Note**
>
> In MQTT, it is possible to quickly update a fleet of gateways with a single `updateFirmware` command. To do this, the gateways must be configured on the same MQTT server and use the same command topic.

`updateLibrary`

SMS/Modbus Slave Command Format

```
updateLibrary
```

Settings None

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.updateLibrary",
```

```
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`log`

SMS/Modbus Slave Command Format

```
log=<interface>:<duration>
```

Settings

Last Name	Required	Description
interface	Yes	`Ethernet`, `tic`, `Serial 1`, `Serial 2`, `Serial 3`
duration	Yes	Duration in minutes

SMS/Modbus Slave Command Example

```
log=serial1:5
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.log",
        "parameters":{
                "interface":"serial1",
                "duration":2
        },
        "callerId":"672"
}
```

`json` Response Format

```
{
        "callerId":"672",
        "result":"OK"
}
```

`setRelay`

SMS/Modbus Slave Command Format

```
setrelay=<action>
```

Settings

Last Name	Required	Description
action	Yes	`open`, `close`, `pulse`
duration	no	only with `pulse`: value in seconds for relay deactivation

> **Note**
>
> The SMS command with the `pulse` action is set to a duration of 1 second

SMS/Modbus Slave Command Example

```
setrelay=pulse
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.setRelay",
        "parameters":{
                "action":"pulse",
                "duration":20
        },
        "callerId":"672"
}
```

`json` Response Format

```
{
        "callerId":"672",
        "result":"OK"
}
```

`setKey`

SMS/Modbus Slave Command Format

```
setkey=<url[:port]>:<login>:<password>:<interface>
```

Settings

Last Name	Required	Description
url	Yes	URL with optional port included (`https://server[:port]`)
login	Yes	Log In
password	Yes	Password
interface	Yes	`Ethernet` or `Modem`.

SMS/Modbus Slave Command Example

```
setkey=ftp://ftp.webdyn.com:2222/script_key.json:modem:login:pwd
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.setKey",
        "parameters":{
                "url":"ftp://ftp.webdyn.com/script_key.json",
                "interface":"ethernet",
                "login":"login",
                "password":"pwd"
        },
        "callerId":"203"
}
```

`json` Response Format

```
{
        "callerId":"203",
        "result":"OK"
}
```

`deleteKey`

This command allows you to remove keys used to decrypt client-side Lua scripts.

> **Note**
>
> If scripts in the `.luax` format are present after the decryption keys have been deleted, they will continue to run as long as the script remains active and the gateway has not been restarted. It is strongly recommended that you delete the `.luax` scripts after deleting the keys.

SMS/Modbus Slave Command Format

```
deletekey
```

Settings None

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.deleteKey",
        "callerId":"203"
}
```

`json` Response Format

```
{
        "callerId":"203",
        "result":"OK"
}
```

NETWORK

`apn`

SMS/Modbus Slave Command Format

```
apn=<apn>:<login>:<password>
```

Settings

Last Name	Required	Description
apn	Yes	Name of the APN
login	No	Camera Connection
password	No	APN Password

SMS/Modbus Slave Command Example

```
apn=m2mapn:login:webdyn
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.apn",
        "parameters":{
                "apn": "iot.1nce.net",
                "login":"",
                "password":""
        },
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`ftp`

SMS/Modbus Slave Command Format

```
ftp=<server>:<login>:<password>:<port>:<interface>
```

Settings

Last Name	Required	Description
server	Yes	FTP Server Address
login	Yes	Log In
password	Yes	Password
port	Yes	FTP Port
interface	Yes	`Ethernet` or `Modem`.

SMS/Modbus Slave Command Example

```
ftp=ftp3.webdyn.com:login:password:21:modem
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.ftp",
        "parameters":{
                "server": "ftp3.webdyn.com",
                "login":"login",
```

```
                "password":"password"
                "port":21,
                "interface":"modem"
        },
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`sftp`

SMS/Modbus Slave Command Format

```
sftp=<server>:<login>:<password>:<port>:<interface>
```

Settings

Last Name	Required	Description
server	Yes	SFTP Server Address
login	Yes	Log In
password	Yes	Password
port	Yes	SFTP Port
interface	Yes	`Ethernet` or `Modem`.

SMS/Modbus Slave Command Example

```
sftp=sftp.webdyn.com:login:password:22:modem
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.sftp",
        "parameters":{
                "server": "sftp.webdyn.com",
                "login":"login",
                "password":"password",
                "port":22,
                "interface":"modem"
        },
```

```
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`https`

SMS/Modbus Slave Command Format

```
https=<server>:<login>:<password>:<port>:<interface>
```

Settings

Last Name	Required	Description
server	Yes	HTTPS/WebDAV Server Address
login	Yes	Log In
password	Yes	Password
port	Yes	HTTPS Port
interface	Yes	`Ethernet` or `Modem`.

SMS/Modbus Slave Command Example

```
https=webdav.webdyn.com:login:password:443:modem
```

SMS/Modbus Slave Format Response

- Success: no return
- Error: Explanatory message

`json` command file format

```
{
        "rpcName":"sunpm.https",
        "parameters":{
                "server": "webdav.webdyn.com",
                "login":"login",
                "password":"password",
                "port":443,
                "interface":"ethernet"
        },
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

MONITORING

`discoverDevices`

SMS/Modbus Slave Command Format

```
discoverdevices=<protocol>:<maxDevices>:<interface>:<timeout>:<port>
```

Settings

Last Name	Required	Description
protocol	Yes	Device protocol (SunSpec, Huawei, SMA, SolarEdge, etc.)
maxDevices	Yes	Maximum number of devices
interface	Yes	`ethernet1`, `ethernet2`, `serial1`, `serial2`, `serial 3`
waiting period	Yes	Wait time in ms
port	Yes	TCP Port

SMS/Modbus Slave Command Example

```
discoverdevices=sunspec:10:serial1:10000
```

SMS/Modbus Slave Format Response

- Success: Number of devices detected
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.discoverdevices",
        "parameters":{
                "protocol":"sunspec",
                "maxDevices":5,
                "interface":"ethernet1",
                "timeout":1000,
                "port":6000
        },
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`getParameters`

SMS/Modbus Slave Command Format

```
getparameters
```

Settings None

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.getParameters",
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`getData`

SMS/Modbus Slave Command Format

```
getdata
```

Settings None

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.getData",
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

INSPECTION

`writeVariable`

SMS/Modbus Slave Command Format

```
writevariable=<deviceName>:<tagName>:<value>
```

Settings

Last Name	Required	Description
Device Name	Yes	Device ID
label name	Yes	Variable/Parameter Label
value	Yes	Enter a value.
		For character strings, use the delimiter "".
		For numeric values, e.g., 12.5

> **Note**
>
> The value is the functional value; the gross value to be entered in the destination register will be calculated automatically based on **CoeffA** and **CoeffB**

SMS/Modbus Slave Command Example

Example with the string for `value`

```
writevariable=INV1:name:"inverter1"
```

Example with an integer for `value`

```
writevariable=INV1:setLimit:30
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error

`json` command file format

```
{
        "rpcName":"sunpm.writeVariable",
        "parameters":{
                "deviceName":"Huawei1",
                "tagName":"Watts",
                "value":77
        },        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

`startScript`

SMS/Modbus Slave Command Format

```
startscript=<name>
```

Settings

Last Name	Required	Description
name	Yes	Script Name

SMS/Modbus Slave Command Example

```
startscript=ActivePowerRegulation-V1_03
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.startScript",
        "parameters": {
                "name": "ActivePowerRegulation-V1_03"
        },
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
```

```
        "result":"OK"
}
```

`stopScript`

SMS/Modbus Slave Command Format

```
stopscript=<name>
```

Settings

Last Name	Required	Description
name	Yes	Script Name

SMS/Modbus Slave Command Example

```
startscript=ActivePowerRegulation-V1_03
```

SMS/Modbus Slave Format Response

- Success: No message
- Error: error message

`json` command file format

```
{
        "rpcName":"sunpm.stopScript",
        "parameters": {
                "name": "ActivePowerRegulation-V1_03"
        },
        "callerId":"1"
}
```

`json` Response Format

```
{
        "callerId":"1",
        "result":"OK"
}
```

7.2.2 Remote Controls

Principle

The gateway can receive remote commands. These commands are used to perform tasks for the following purposes:

- configuration
- control
- monitoring

Here are a few examples:

- Search for equipment
- Get the current configuration
- initiate a connection to the servers
- call a function from a script installed on the gateway

An order can be placed using four different methods:

- A command file stored on the server (**FTP, SFTP ou WebDAV**) that will be retrieved by the gateway when it connects to the server
- A message posted by **MQTT** in the WebdynSunPM monitoring thread
- A **SMS** sent to the WebdynSunPM SIM card
- A Modbus device **Master** with the **Modbus Slave** function enabled

List of External Commands

Orders	Category	Description
connect	System	Initiating a connection
status	System	Retrieving the gateway's status
factory	System	Restore Factory Settings
reboot	System	Restarting the gateway
updateFirmware	System	Gateway Software Update
updateLibrary	System	Library Update
log	System	Enabling Communication Logs for Devices
setKey	System	Add keys for decrypting client scripts
deleteKey	System	Removal of keys used to decrypt client scripts
setRelay	System	Changing the Relay Status
apn	Network	Modem Setup
ftp	Network	FTP Server Configuration
sftp	Network	SFTP Server Configuration
https	Network	Configuring the WebDAV/HTTPS Server
discoverDevices	Monitoring	Exploring Facilities
getParameters	Monitoring	Collecting parameters—that is, variables defined with action code 1
getData	Monitoring	Collecting variables defined with action code 6 or 7
writeVariable	Control	Writing a Variable to a Device
startScript	Control	Running a Script
stopScript	Control	Stopping a script

Command File (FTP, SFTP, WebDAV)

When an FTP, SFTP, or WebDAV connection is established, the gateway checks for the presence of a command file in the directory designated for this purpose (`/CMD` by default). This file must be named `<UID>_cmd.json`, where `<UID>` is the gateway ID.

The commands included in this file are listed in **format JSON** below and are executed in order. Calling script functions is also supported.

Note

The file is deleted after import to prevent the same order from being processed twice.

Command results are also written to files that will be submitted during the next connection. A file may contain one or more results. These are named according to the `<UID>_ACK_<timestamp>.json` pattern.

Note

Only the main server (Server 1) is used.

JSON Format of the Command File

```
[
  {
    "rpcName": "<nom du script>.<nom de la fonction>",
    "parameters": {
      "<cle_parametre>": "<valeur>"
    },
    "callerId": "<identifiant commande 1>"
  },
  {
    "rpcName": "<nom du script>.<nom de la fonction>",
    "parameters": {
      "<cle_paramètre>": "<valeur>"
    },
    "callerId": "<identifiant commande 2>"
  }
]
```

- **rpcName**: REQUIRED - Name of the script and function to be executed in the format `<nom du script>.<nom de la fonction>`. For a command, use "sunpm" as the script name, which is reserved for WebdynSunPM's internal commands.
- **paramters**: OPTIONAL - Some functions and commands require additional parameters. When this is not the case, this field is optional.
- **callerId**: OPTIONAL - An identifier associated with this request.

Note

Each entry in the table corresponds to a different command. Note that if the file contains only one command, the square brackets are optional.

JSON format of the results file

The format of the output file is as follows:

```
[
  {
    "result": {
      "<clé_retour>": "<valeur>"
    },
    "error": null,
    "callerId": "<identifiant commande 1>"
  },
  {
    "result": null,
    "error": "<description de l'erreur>",
```

```
    "callerId": "<identifiant commande 2>"
  }
]
```

- **result**: If the function or command is successful, it returns a result in JSON format contained in this field. If an error occurs, this field is empty.
- **error**: If an error occurs, this field contains a description of the problem encountered. If the function or command is successful, this field is empty.
- **callerId**: The same identifier as in the request. This allows you to associate this response with its original request.

Example

Let's start with the following example script to illustrate how to call a function:

```
header = {
        version = 1.1,
        label = "Test",
        name = "test"
}
--[[
        Test function
]]
function testFunction(parameters)
        wd.log("question is " .. parameters.text)
        local result = {
                value = 42
        }
        return result
end
```

Below is a command file that can be used to call the function:

```
{
        "rpcName":"test.testFunction",
        "parameters": {
                "text":"what is the answer ?"
        },
        "callerId":"3d9311ed-0076-4f28-ac59-a2debfa35b86"
}
```

And here is the resulting file:

```
{
        "result": {
                "value":42
        },
        "callerId":"3d9311ed-0076-4f28-ac59-a2debfa35b86"
}
```

MQTT Command Message

The MQTT server must be configured on Server 2. In particular, the command and result topics must be specified.

The gateway subscribes to the command topic so that any message sent to it is received and executed. The result of a command or function call is published to the result topic.

MQTT Command Format

An MQTT message can contain only one command or result; the format is as follows:

```
{
    "result": {
      "<clé_retour>": "<valeur>"
    },
    "error": null,
    "callerId": "<identifiant commande 1>"
}
```

> **Warning**
>
> Unlike an SFTP/FTP/WebDAV server, which can contain multiple commands with `[ ]`, MQTT does not allow this

Azure Iot

> **Warning**
>
> Azure has its own function invocation mechanism. The JSON format used is therefore as follows:

```
{
        "methodName":"<nom du script>.<nom de la fonction>",
        "responseTimeoutInSeconds":"<délai avant timeout>",
        "payload":{
                "parameters":{ "<paramètres de la fonction au format json>" }
        }
}
```

- **methodName**: replaces rpcName
- **parameters**: content in the payload field of the Azure message
- **callerId**: ignored

Text Message

The gateway can receive and send text messages. To do so, you must have a mobile connection that is configured and activated.

> **Note**
>
> **A partir du FW 5.1.5**
>
> Text messages received by the gateway are encrypted to ensure the security of the transmitted information. Responses from the gateway are not encrypted.

SMS Encryption

The gateway uses OpenSSL's **l'AES-256-GCM+PBKDF2** encryption. The gateway comes with a default encryption key, which is printed on the product label.

The encryption key can be changed at any time.

> **Warning**
>
> A factory reset restores the SMS encryption settings to the factory defaults.

An implementation note is available that details the encryption method.

In addition, Webdyn provides a web page that allows you to encrypt text messages:

Page de chiffrement SMS - WebdynSunPM

> **Note**
>
> This webpage allows you to encrypt text messages but does not allow you to send them

SMS Command Format

The SMS format uses a simplified format:

```
<commande 1>=<paramètre 1>:<paramètre 2>:<paramètre 3>… ;
<commande 2>=<paramètre 1>:<paramètre 2>:<paramètre 3>…
```

You can send multiple commands by adding the separator `;` ; the last command must not end with `;`

Example

Here is an example of an SMS command:

Avant chiffrement

`connect=1`

Après chiffrement

`xen21/1YOId7NjxxYIj2/HmFRPiONoxmHtkrJ/ZqaZPzO+/NjgrUIriQjsUpQVPLow2rdtOlY1uAVo=`

Modbus TCP Slave

It is possible to execute commands in Modbus slave mode.

Modbus Slave TCP Command Format

Simply write a string of characters to the registry key `34000` similar to the unencrypted SMS command:

```
<commande 1>=<paramètre 1>:<paramètre 2>:<paramètre 3>…
```

Note

It is not possible to place multiple orders at once

Note

An order cannot exceed 120 characters

Warning

If there is an error in a submitted order, no error message will be returned