TITAN

User Manual
Software

Index

General Notes

The product is deemed to have been accepted by the recipient and is provided without an interface for the recipient's products. The documentation and/or the product are provided for testing, evaluation, integration and information purposes. The documentation and/or products are provided "as is" and may include defects. The documentation and/or products are provided without a warranty of any kind, either express or implied. To the fullest extent permitted by the applicable laws, Webdyn further disclaims all guarantees; including, but not limited to, all implied guarantees of merchantability, integrity, fitness for a particular purpose, and non-infringement of third-party rights. All risks arising out of the use or performance of the product, or the documentation are borne by the recipient. This product is not intended for use in life support devices or systems where a malfunction of the product can reasonably be expected to result in personal injury. Applications incorporating the described product must be designed in accordance with the technical specifications provided in these guidelines. Failure to follow any of the required procedures may result in a malfunction or serious discrepancies in the results.

Furthermore, all safety instructions related to the use of mobile technical systems, including GSM products, which also apply to cell phones, must be strictly followed. Regardless of the legal theory on which a claim may be based, neither Webdyn nor its suppliers shall be held liable for any consequential, incidental, direct, indirect, punitive or other damages (including, without limitation, damages for lost profits, interruption of business, loss of business data or information, or other pecuniary losses) arising from the use, or inability to use, the documentation and/or the product, even if Matrix Electronics has been advised of the possibility of such damages occurring. The foregoing limitations of liability shall not apply in the event of mandatory liability, e.g. pursuant to the Spanish Product Liability Law, or in the event of intent, gross negligence, injury to life, body and health, or breach of a condition in relation to the contract. However, claims for damages arising from a breach of a condition relating to the contract, shall be limited to the foreseeable damage which is intrinsic to the contract, unless caused by intent or gross negligence, or is based on the liability for injury to life, body and health. The provision does not imply a change in the burden of proof to the detriment of the recipient. Subject to change without notice. The interpretation of this general note will be governed and interpreted in accordance with Spanish law, without reference to any other substantive law.

Important Information

This technical description contains important information for the start-up and use of TITAN-based devices. Please read it carefully before you start working with them. The warranty will be void if damage occurs due to non-compliance with these instructions. We cannot accept liability for related losses.

Revisions

VERSION. 6.25

1.- Introduction.

TITAN firmware enables 2G/3G/4G devices with advanced Gateway capabilities.

In addition to the classic router capabilities, TITAN-based devices can create 4G/3G/2G - RS232/485 gateways, execute AT commands via SMS (to check coverage, switch relays, read digital inputs, etc.), accept GSM data calls (CSD) to access devices such as electricity meters, autonomously read Modbus RTU or Modbus TCP devices, send temperature and distance data to the cloud, and much more.

To better understand what you can do with these devices, we recommend you read chapter 2 of this manual, the FAQ section. After reading all this information, you will have a much better idea of what you will be able to do. After reading the FAQ section, we recommend you take a look at the application notes (PDF documents external to this document). These examples will give you a much clearer picture of the possibilities offered by these devices.

We provide free, fast and efficient support to all users of modems and routers when required. Therefore, if you still have any doubts or questions after reading this manual, please feel free to contact us at iotsupport@mtxm2m.com. Similarly, if you need a feature which is not included in our devices, or if you need a special customisation, please let us know and we will perform an in-depth study on it.

2.- FAQ. Basic concepts.

- **What can I do with TITAN-based devices?**

Essentially, you can do exactly the same things as with a standard 4G/3G/2G router, but with the addition of a number of advanced gateway features. For example, they can provide Internet connectivity to devices that are connected to their Ethernet port, as well as perform NAT to connect via the Internet to devices connected to the Ethernet port.

- **Is it possible to provide Internet connectivity to WiFi-enabled devices?**

Effectively, TITAN-based devices with WiFi connectivity can work as a WiFi Access Point, and thus, can provide Internet connectivity to WiFi client devices that are connected to it. They can also be configured as a client WiFi, i.e., the TITAN-based devices can connect to an existing WiFi Access Point in your infrastructure and use this connection as internet access.

- **Can you have serial gateways?**

Yes, you can have as many 4G/3G/2G-Serial gateways as there are serial ports on the devices you are using. These gateways can either be of type TCP Client or TCP Server and can all work simultaneously.

- **RS232 or RS485 serial gateways?**

All devices have at least one of or both RS232 and RS485 serial ports. Depending on the type of serial port, the gateways will be of one type, the other or both.

TITAN-based devices can be used to create IP - RS232/485 gateways, but can these be SSL/TLS-secure?

Yes, you can use TITAN-based devices to configure IP - RS232/485 gateways of Server type and Client type with SSL/TLS security, with mutual authentication by digital certificate (client/server).

- **Can you also take CSD calls using TITAN firmware? So, can I have a 4G/3G/2G-Serial gateway in operation while also accepting a GSM/CSD data call for an electricity meter reading?**

Yes, but with conditions. If the TITAN-based device is operating in 2G mode, there won't be any issues. The device will accept the CSD call when the call is received (to perform a CSD-Serial gateway) and will take over the 2G-Serial gateway at the end of the call. If the device is also configured to work in 3G or 4G mode, whether the CSD/GSM call is accepted will depend on your telephone provider, as not all operators will accept CSD/GSM calls. For example, if you use Movistar or Orange and have the device operating in "auto

4G/2G" mode, there will be no problems accepting the CSD call, regardless of whether the TITAN-based device is operating in 2G or 4G mode (in 4G, it will fail over to 2G), but this will not happen if you are with Vodafone, because it isn't supported.

- **I want a 4G/3G/2G router, but I need to be able to send AT commands directly to TITAN-based devices, in order to send SMS messages, check coverage, etc. Do TITAN-based devices allow this?**

Yes, all models and in various ways. It is possible to send AT commands REMOTELY from a "Telnet-like" connection, or via SSH, Modbus TCP, vSNMP, HTTP/HTTPs, MQTT/MQTTs, SMS, and even embedded commands in the IP-Serial gateways themselves. It is also possible to send AT commands LOCALLY through the Ethernet port via "Telnet", SSH, Modbus TCP, SNMP, HTTP/HTTPs and through the RS232/RS485 serial port via embedded AT commands, or Modbus RTU.

- **Do TITAN-based devices support a Modbus TCP gateway to Modbus RTU?**

Yes, they support the conversion of these protocols, so if you need to remotely access Modbus RTU devices, the TITAN-based device will allow you to do so by acting as a Modbus TCP gateway to Modbus RTU.

- **And can TITAN-based devices also operate autonomously by reading Modbus RTU or Modbus TCP devices and sending the readings to the cloud?**

Yes. TITAN-based devices can read Modbus RTU and/or Modbus TCP devices, even in combination, store the readings internally, and send them to WEB platforms through a JSON object via HTTP/HTTPs, MQTT/MQTTS or FTP.

- **What if the TITAN-based device is reading Modbus RTU/TCP devices autonomously? Would that prevent me from connecting sporadically and remotely to the Modbus RTU/TCP devices from my office to make occasional readings, change firmware, etc?**

No. You can configure TITAN-based devices to read Modbus devices autonomously and set up a sporadic direct gateway to access Modbus RTU devices from your premises.

- **Do TITAN-based devices have datalogger capability and can they store data?**

Yes. As mentioned previously, TITAN-based devices can store many types of data for subsequent delivery to web, MQTT and FTP platforms. These include, for example, readings from Modbus RTU/TCP devices, but also temperature readings, generic RS232/RS485 serial data, IEC-102 meter readings, digital inputs and outputs, etc.

- **How do you configure TITAN-based devices?**

There are several ways to configure TITAN-based devices, but generally, all device configurations can be done via web configuration, i.e. via the internal web server of the devices.

- **Is it possible to read/upload a complete configuration to the devices? This facilitates the production process when there are a significant number of devices to configure.**

Yes, it is possible to make full configuration backups / restores (to copy the configuration of a TITAN-based device to another TITAN-based device).

- **Can you order TITAN-based devices pre-loaded with a particular configuration from the factory?**

Yes, you can. But only for high volumes (>500 units). Please contact your sales representative for more information.

- **Do you support the DynDNS service?**

Of course. It is also compatible with NO-IP. You also have the additional option of "Private DNS" to send your current IP every time it changes or periodically to a private server, e.g.: your company's server. As part of this additional option, as well as sending the current IP address, other data such as coverage, technology, IMEI, status of digital inputs and outputs, etc. are attached.

- **So, TITAN-based devices allow you to manage digital inputs and outputs?**

Yes. If your TITAN-based device has digital inputs or outputs, it will be able to manage them. The internal datalogger can be easily configured to store and be sent to a web platform (via HTTP/HTTPS, MQTT/MQTTS or FTP) every time there is a change in a digital input or to set a time period for reading and sending the status of these inputs. It is also possible to query the status of a digital input at any time by AT command (which can be sent by Telnet, SSH, HTTP/S, SMS, MQTT/S, Modbus, SNMP, RS232, RS485, etc.). The same applies to the digital outputs.

You can also access the digital inputs/outputs from the "TITAN Scripts" section of the device, from where it is possible to program a small script (in JavaScript) so that the device uses these digital inputs/outputs at the user's convenience.

- **Do TITAN-based devices have a clock?**

Yes, they have a built-in clock. It is synchronised via NTP over the Internet.

- **Can TITAN-based devices be configured via SMS?**

Yes, TITAN-based devices can be configured through AT commands. And AT commands (in addition to Telnet, SSH, HTTP/S, MQTT/S, Modbus, SNMP, RS232 and RS485) can be sent to the device by SMS. Commands can be sent to the devices to reboot them, change their configuration, find out their IP address, check coverage, etc.

- **Is it possible to customise the web configuration environment with company logos and images?**

Yes. Any user is free to customise the configuration web environment with their logos, footers and page titles, etc. It is even possible to choose which menus your end customer can see or modify.

- **Do TITAN-based devices have an indicative LED?**

Yes, the number of LEDs depends on the model. You can choose the functionality of the LEDs to indicate coverage, if the device has an IP, SIM problems, etc. You can also use the LEDs from the TITAN Scripts,

where it is possible to program a small user JavaScript script to turn a device LED on or off depending on needs (for example, when a digital input is activated).

- **Can you use a TITAN-based device as a serial datalogger?**

Yes. A TITAN-based device can be configured to read everything that arrives on the serial port, store it in its internal datalogger and send it via HTTP/S, MQTT/S or FTP to a platform over the Internet.

- **And can TITAN-based devices be used to read IEC 870-5-102 meters?**

Yes. With TITAN-based devices, you can do the following with regard to reading electricity meters with the IEC 870-5-102 protocol:

- 1.- Meter reading via a 4G/3G/2G-RS232/485 gateway.

- 2.- Meter reading via a CSD data call.

- 3.- Configure the TITAN-based device to read the meter autonomously during each configured time period, sending the instantaneous values to a web platform (via HTTP/S, MQTT/S, or FTP) during each time period and also, if desired, sending pricing information once a day.

- 4.- In the "TITAN scripts" section, you can program a small JavaScript script to read the electricity meter however you wish (e.g.: constantly monitor a certain variable, obtain more accurate average values than the quarter-hour readings, etc.).

- Is it possible to set up a VPN using TITAN-based devices?

Yes. TITAN-based devices allow you to configure an IPSEC or OpenVPN VPN.

- **What other relevant features do TITAN-based devices have?**

Throughout this manual you will find detailed descriptions of the various features, but in addition to what has already been indicated in the previous paragraphs, you can also use TITAN-based devices for the following:

- Auto-reset configuration (periodic auto-reset, at a certain time, auto-reset in case of IP connectivity problems, etc.)

- Authentication service by Tacacs+ for HTTP, Telnet and SSH

- Secure configuration via HTTPS

- User permissions configuration to enable access to the various functionalities of the TITAN-based equipment, according to the "admin", "user" and "guest" roles.

- SMTP service configuration for sending emails

- Configuration as SLAVE MODBUS. This allows TITAN-based devices to be configured and/or read via Modbus. For example, a TITAN-based device can be used to read the status of its digital inputs via Modbus,

or to obtain the time of the device, change a digital output, etc., all from a Modbus Master device such as a PLC.

- TITAN scripts. This section allows you to write a small user script in JavaScript to, for example: read serial devices or send custom data to them, read Modbus devices, read or change a digital output, send SMS, EMAILs, SNMP traps, read an IEC 870-5-102 meter, send MQTT messages, perform PINGs, change the status of the device's LEDs, write and read files, send or receive files via FTP, etc.

3.- Step-by-Step Configuration.

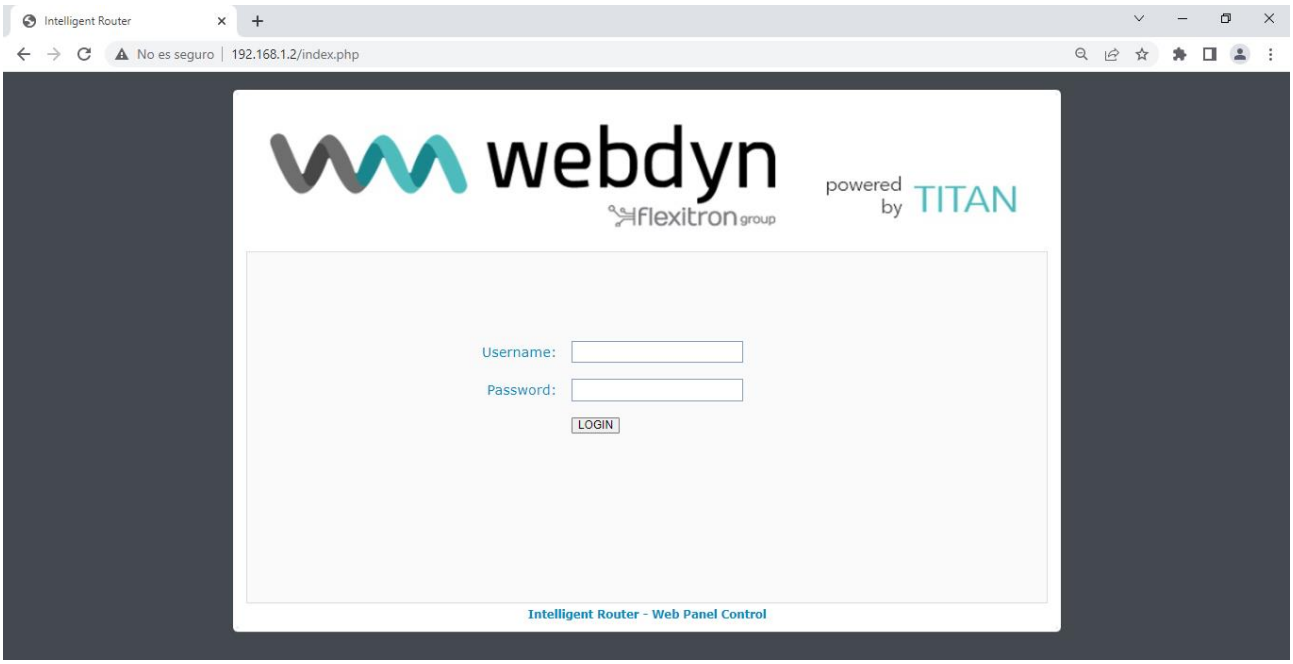
As mentioned in previous sections, the configuration of TITAN-based devices is mostly carried out through their web-based environment.

What is required?

- A PC with a web browser (Chrome, IExplorer, Firefox, etc.) and an Ethernet port.
- An Ethernet cable to connect the PC to the device.

Steps to access the configuration environment.

- Connect an Ethernet cable between the PC and device.
- The PC must have a fixed IP address, within the range 192.168.1.X, as the IP address is 192.168.1.2
- Open a browser with the address http://192.168.1.2. The following window should appear:



- Use the default username and password: admin and admin

4.- Configuration

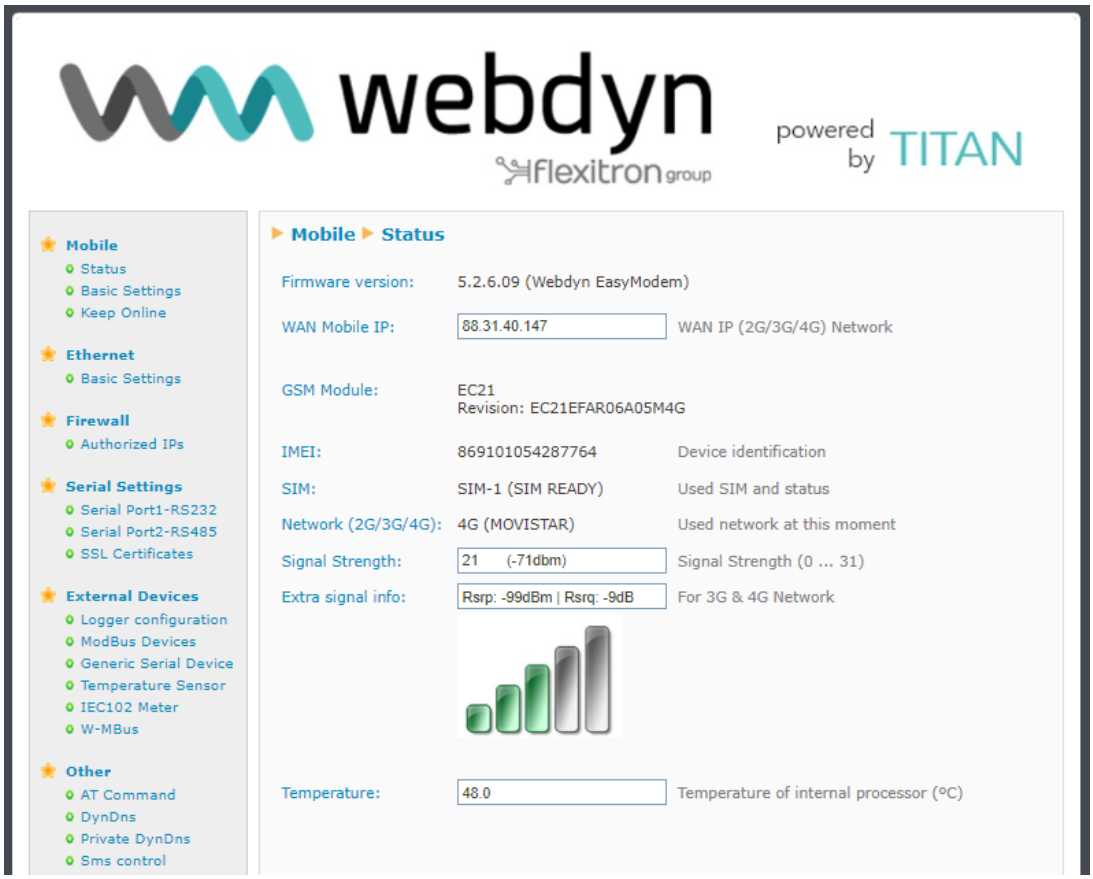
4.1 - Mobile

The Mobile section covers all aspects related to the device's 2G/3G/4G configuration, including the connection status, network configuration parameters and connection monitoring.

4.1.1 - Mobile → Status

This screen shows the general status of the device.

- Firmware Version: FW version of the device
- ***WAN Mobile IP***: WAN IP address (IP address assigned to the connection (2G/3G/4G) if available).
- ***GSM Module***: indicates the manufacturer and model of the internal GSM module of the device.
- ***Network (2g/3g/4g)***: indicates whether the current WAN connection is using the 2G (GPRS), 3G or 4G network.
- ***Signal Strength***: indicates the strength of the signal. 0=none, 31=maximum
- ***Extra signal info:*** indicates RSCP/ECN0 and RSRP/RSRQ for 3G and 4G connectivity.
- Internal Temperature: displays the internal temperature of the processor.



4.1.2- Mobile → Basic Settings

This section covers the configuration of the WAN connection (4G/3G/2G). parameters. You will need to know about your SIM card, including the APN, username and password. Your provider must give them to you.

- ***Mobile WAN***: if you need IP connectivity (4G/3G/2G), set this option to "Enabled". If you are not going to use a SIM card, set it to "Disabled".
- ***Sim Mode***: Depending on the device model, 1 SIM slot or 2 SIM slots are available. If you have 2 SIM slots: "SIM1": the router will use SIM1. "SIM2": the router will use "SIM2". "SIM1 + SIM2 (backup)": the router will use SIM1, and in case of problems will use SIM2.
- ***SIM1 APN***: APN of the SIM1 operator. Ask your GSM provider.
- ***SIM1 Username***: username of the SIM1 operator. Ask your GSM provider.
- ***SIM1 Password***: SIM1 operator password. Ask your GSM provider.
- ***Sim1 Pin***: if your SIM1 card has a PIN, you must enter it here.
- ***SIM1 Auth****:* authentication method "NONE, PAP", CHAP , AUTO (PAP or CHAP)".
- ***SIM2 APN***: APN of the SIM2 operator. Ask your GSM provider.
- ***SIM2 Username***: username of the SIM2 operator. Ask your GSM provider.
- ***SIM2 Password***: SIM2 operator password. Ask your GSM provider.
- ***Sim2 Pin***: if your SIM2 card has a PIN, you must enter it here.
- ***SIM2 Auth****:* authentication method "NONE, PAP, CHAP , AUTO (PAP or CHAP)".
- ***Network selection:***

Auto (4G/3G/2G): the device will use 4G if there is coverage, or 3G or 2G otherwise.

Auto (4G/2G): the device will use 4G if there is coverage, or 2G otherwise.

4G: the device will use the 4G network in all cases. If there is no 4G coverage, it will not switch to 2G or 3G

3G the device will use the 3G network in all cases. If there is no 3G coverage, the device will not switch to 2G.

2G: the device will use the 2G network in all cases.

- ***DNS selection:***

"Get DNS from Operator" makes the DNSs used by the device the ones assigned by the telephone operator.

"Selected DNS servers" makes the DNS servers used the ones specified in the DNS1 and DNS2 parameters discussed below.

- ***DNS1 and DNS2**:* DNS servers for domain name resolution. We recommend using Google's 8.8.8.8 and 8.8.4.4, or those indicated by your provider. These DNSs will be those used if the FailOver feature is selected when the data output is produced via the 4G/3G/2G WAN interface

***Remote webserver management**:* if you check the box, you will be able to access the web configuration page of the device remotely, through its public IP address (the one indicated in Mobile → Status). You can change the port from the Other→Http / Https menu

Additional Notes.

- Once the configuration process is finished, click on the "SAVE CONFIG" button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.1.3 - Mobile → Keep Online

On this screen you can configure a PING to check the device's connectivity. If the PING fails in the configured instances, the 4G/3G/2G session will be restarted. We recommend you use this feature.

- ***Enabled***: check the box to allow the TITAN-base device to send a PING in order to check connectivity periodically.
- ***Ping Server***: indicates the IP or DNS address of the server to PING.
- ***Period***: indicates the number of minutes between each PING
- ***Timeout***: indicates the number of seconds the device must wait for the PING response before deeming that the PING has failed.
- ***Retries***: indicates the number of PING retries that must be performed before considering connectivity issues and restarting the 4G/3G/2G session.
- ***Retry period***: indicates the number of minutes between retries.

Additional Notes.

- Once the configuration process is finished, click on the "SAVE CONFIG" button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.1.4- Mobile → Basic Settings → FailOver

This configuration screen can be accessed by clicking on the button at the bottom of the Mobile → Basic Settings section.

Configuring the device's failover characteristics. For example, to configure the data output of the TITAN-based device via its WiFi or Ethernet interface and, in the event of an Ethernet or WiFi interface failure, bring up the 4G/3G/2G interface so that data output occurs over that interface until the WiFi or Ethernet interface regains connectivity.

- ***Enabled***: check the box to activate the failover functionality.

- ***IP1***: to check the connectivity of the failover interface used (Ethernet or WiFi), the TITAN-based device periodically performs a PING on this IP. In case of failure, the ping is reattempted on IP2.

- ***IP2***: to check the connectivity of the failover interface used (Ethernet or WiFi), the TITAN-based device performs a PING on this IP2 if IP1 fails. In case of failure, the WAN interface (4G/3G/2G) is activated for use for the duration of the connectivity failure.

- ***Interface***: communications interface (Ethernet or WiFi) used as the primary data output interface.

- ***4G/3G/2G interface***: 4G/3G/2G backup interface behaviour. It is possible to choose between "On when failover" or "Always on" mode. In "On when failover" mode the 4G/3G/2G WAN interface is switched off, and only when the primary interface fails (WiFi or Ethernet interface) is it activated and later used. In "Always on" mode, the 4G/3G/2G WAN interface is always active and it is used when the primary interface (WiFi or Ethernet) fails. As the 4G/3G/2G WAN interface is already active, the switchover happens more quickly.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

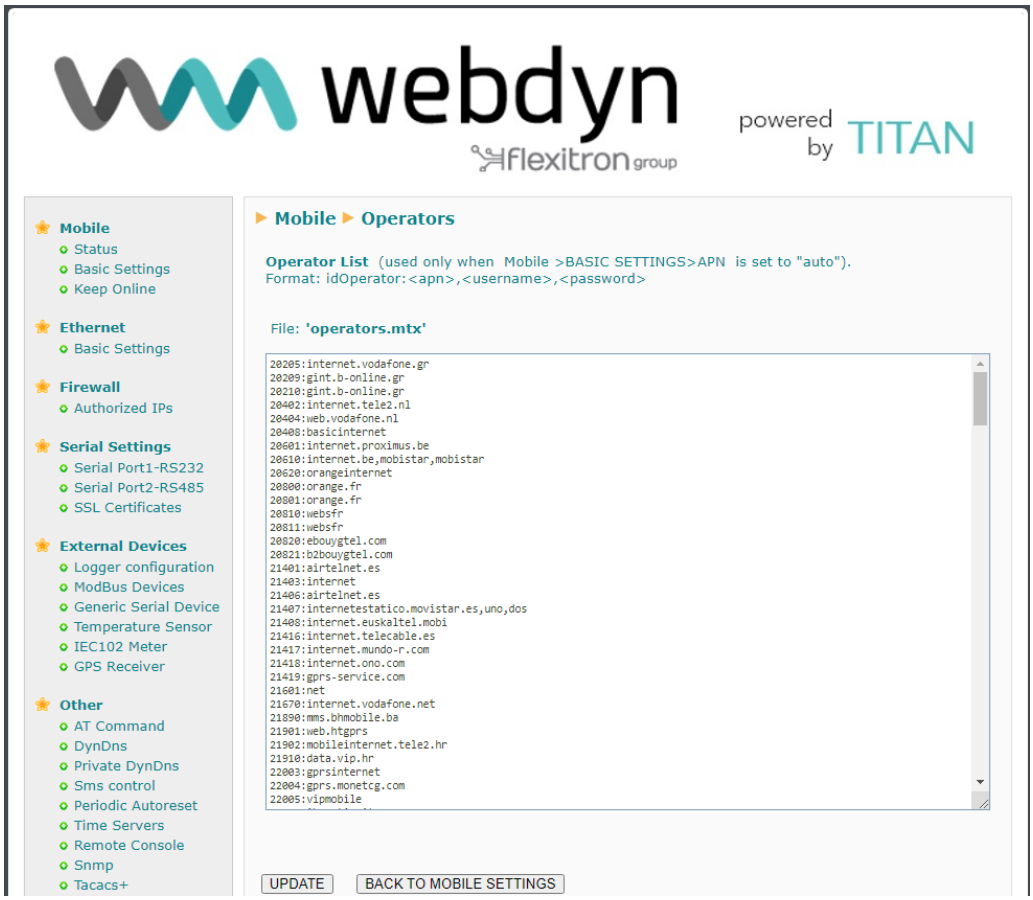
4.1.5- Mobile → Basic Settings → Operators

This configuration screen can be accessed by clicking on the button at the bottom of the Mobile → Basic Settings section.

This allows you to configure the APN, user and password to be used if the APN is automatic. You can use the APN in automatic mode by specifying the "auto" text (without quotation marks) in the APN fields in the Mobile →Basic Settings section.

The "operators.mtx" file must be in the following format:

<ID_Operator>:<apn>,<username>,<password> (where <username>,<password> are optional)



Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.1.6- Mobile → Basic Settings → SIM Swapping

This configuration screen can be accessed by clicking on the “SIM swapping settings” button at the bottom of the Mobile → Basic Settings section. This configuration will only be available on devices with a DUAL SIM.

- ***Main SIM***: If you have selected a Dual SIM configuration in the "Mobile → Basic settings” menu, i.e. a "SIM1 + SIM2" configuration, this menu will allow you to select the main SIM, that is, the first SIM to come into operation after resetting the Titan router.

- ***Swap by network***: allows you to select a time in seconds in which the SIM swap process will be initiated if the Titan router is not able to register on the network within the specified time (in seconds). A value of "0" indicates that this feature will not be used. The minimum value (value used by default) is "30" seconds and the maximum value is "1800" seconds.

- ***Swap by IP***: allows you to select a time in seconds in which the SIM swap process will be initiated if the Titan router is not able to obtain a network IP address within the specified time (in seconds). A value of "0" indicates that this feature will not be used. The minimum value (default value used) is "90" seconds and the maximum value is "1800" seconds.

- ***Swap by coverage***: allows you to select a time in seconds in which the SIM-card swap process will be initiated if the Titan router is still in the process of connecting to the network, but with critical

coverage. The minimum value is "600" seconds and the maximum value is "86400" seconds. The default value is "0", when disabled.

- ***Swap by keep alive***: if this option is selected and the Titan router restarts the IP connection following the failure of the connectivity ping of the keep online function (which can be activated from the menu "Mobile → Keep online"), the SIM card will be swapped. This option is disabled by default.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

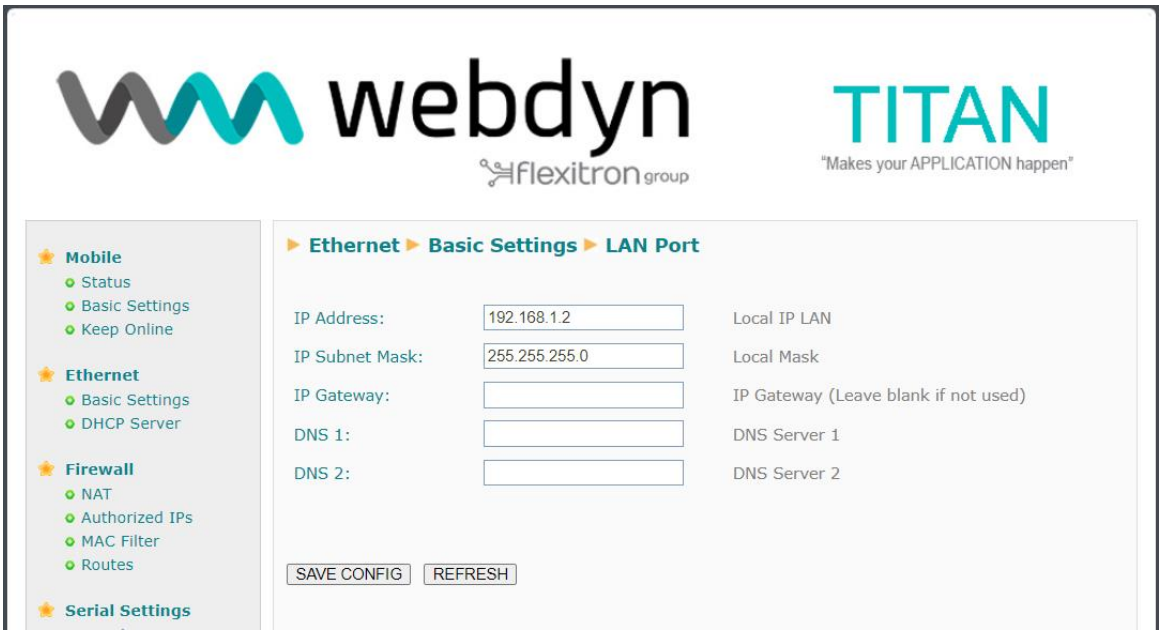
4.2- Ethernet

The “Ethernet” configuration section refers to the local Ethernet port settings for the TITAN-based device.

4.2.1- Ethernet → Basic Settings

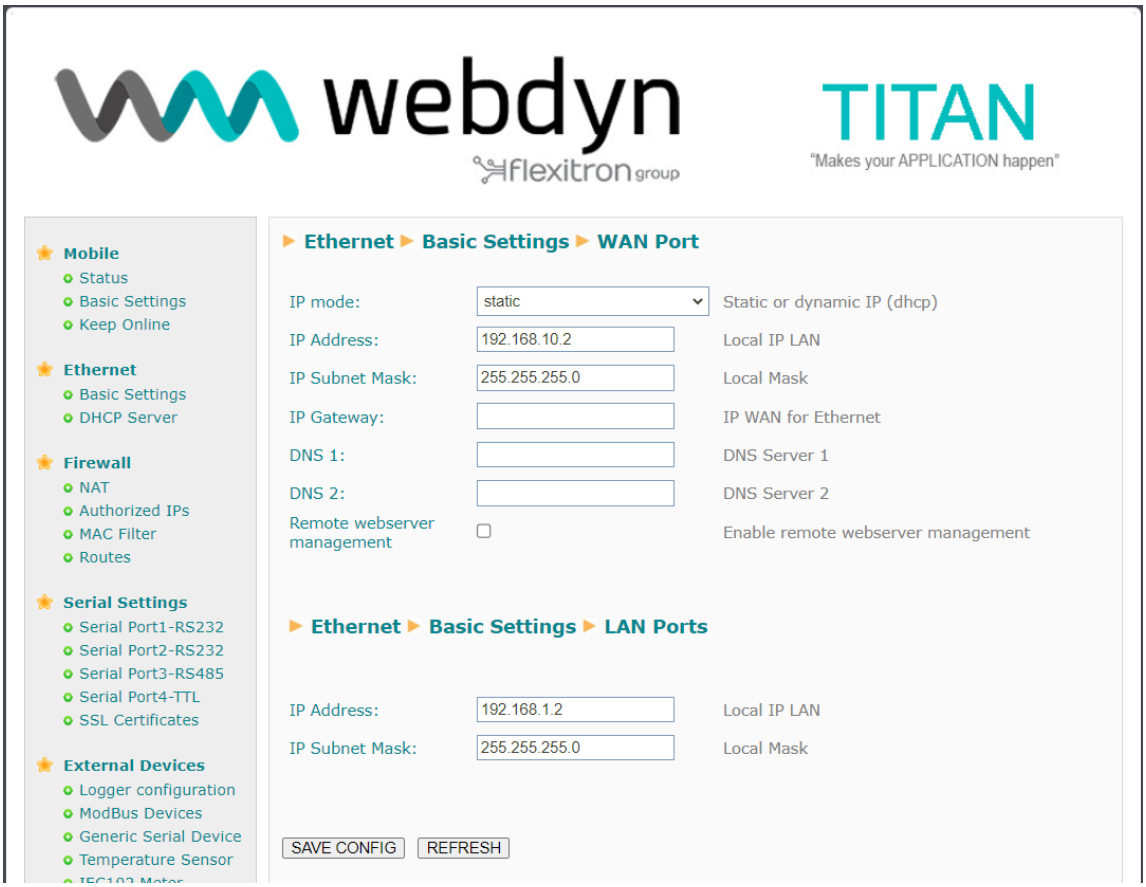
This section lets you configure the basic network parameters of the Ethernet connection. This menu is slightly different depending on whether the TITAN-based device has one or more Ethernet ports.

TITAN-based devices with 1 Ethernet port:



- ***IP Address***: local static IP address of the Ethernet interface (by default, 192.168.1.2)
- ***IP Subnet Mask***: subnet mask (by default, 255.255.255.0)
- ***IP Gateway***: IP address of output gateway. Leave blank if a gateway is not to be used.
- ***DNS1***: Primary DNS server
- ***DNS2***: secondary DNS server

TITAN-based devices with multiple Ethernet ports:



For devices with more than one Ethernet port, there will always be 1 WAN port and the rest will be LAN ports.

WAN port configuration:

- ***IP Mode***: allows you to use a static or dynamic (DHCP) IP configuration for the WAN port.
- ***IP Address***: local static IP address of the WAN Ethernet interface (by default, 192.168.1.2)
- ***IP Subnet Mask***: subnet mask of the WAN Ethernet interface (by default, 255.255.255.0)
- ***DNS1***: primary DNS server
- ***DNS2***: secondary DNS server
- ***IP Gateway***: IP address of output gateway

LAN ports configuration:

- ***IP Address***: local static IP address of the LAN Ethernet interface (by default, 192.168.10.2)
- ***IP Subnet Mask***: subnet mask of the LAN Ethernet interface (by default, 255.255.255.0)

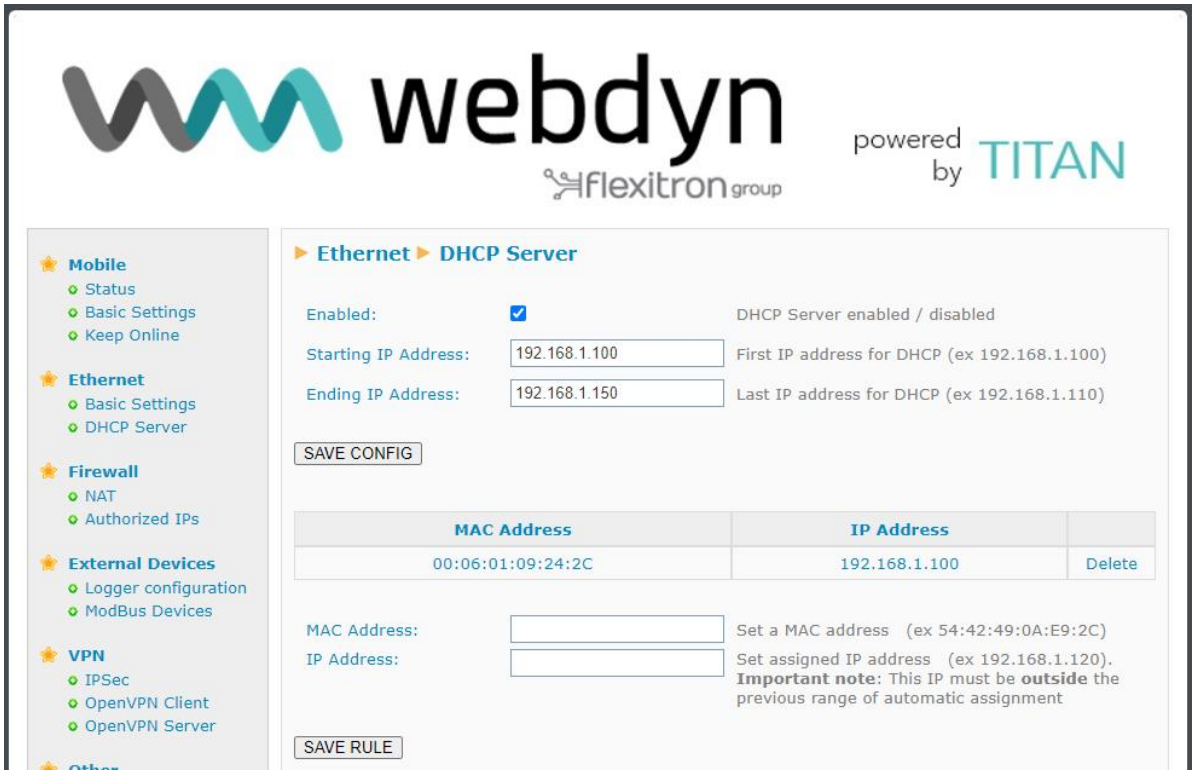
Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.2.2 - Ethernet → DHCP Server

This section allows you to enable and configure the DHCP server assigned to the Ethernet interface of the TITAN-based device.

- ***Enabled***: checking the box will enable the DHCP server on the Ethernet interface.
- ***Starting IP Address***: indicates the first IP address to be assigned by the DHCP server
- ***Ending IP Address***: indicates the last IP address to be assigned by the DHCP server
- ***MAC Address / IP Address****:* these two parameters allow the DHCP server to always assign the same IP address to a given MAC address.



Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Press the “SAVE RULE” button for each MAC / IP pair you wish to create. You can create up to 10.
- Remember that you must restart the device for the new changes to take effect

4.3 - Wifi

The “WiFi” configuration section refers to the local WiFi interface settings. By using a WiFi configuration in "Access Point" mode, you can, for example, provide Internet access to client WiFi devices or configure the TITAN-based device itself. Through a configuration in "WiFi Client" mode, you can also get the TITAN-based device to use another WiFi Access Point on your premises as an Internet connection.

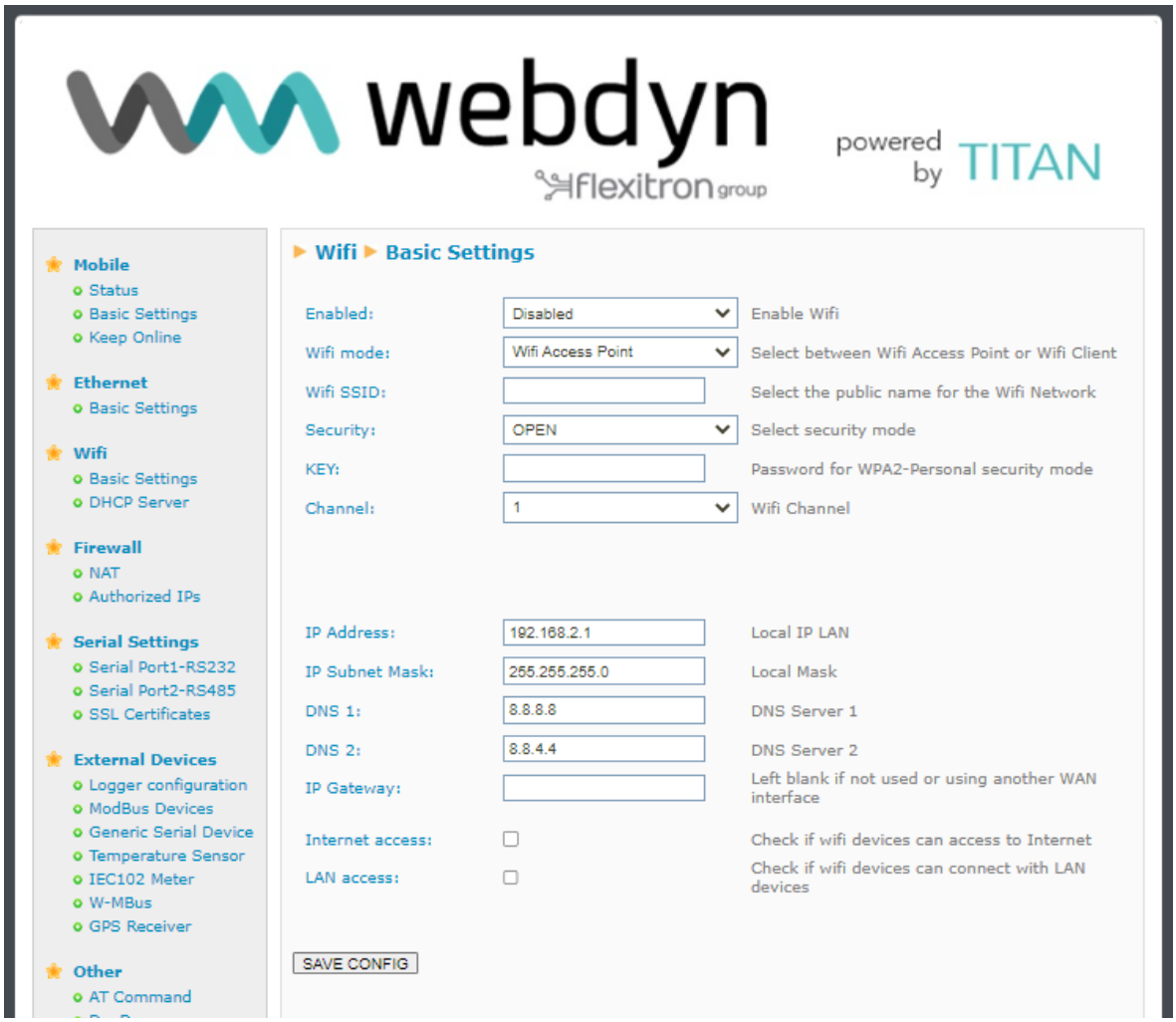
4.3.1 - WiFi → Basic Settings

This section lets you configure the basic network parameters of the WiFi interface.

- ***Enabled***: checking the box enables the WiFi interface of the TITAN-based device.
- ***Wifi mode***: allows you to select between "WiFi Access Point" or "WiFi Client" operating mode. Select the "WiFi Access Point" mode if you wish to use the TITAN-based device to provide Internet access to WiFi-enabled devices using the 4G/3G/2G interface. Select "WiFi Client" if you wish to use an existing WiFi infrastructure (e.g.: a company's fibre router) to connect to the Internet.
- ***WiFi SSID***: Public SSID (indicates the SSID to be created by the TITAN-based device in "WiFi Access Point” mode, as well as the SSID to which the TITAN-based device will be connected in "WiFi Client" mode).
- ***Security***: allows you to specify the WiFi security mode. WPA2 recommended
- ***KEY***: specifies the WiFi password, required for WPA2 mode

- ***IP Mode***: specifies, in "WiFi Client" mode, whether a Local WiFi IP address is to be used as the static IP address or whether it has to take the IP via DHCP.
- ***IP Address***: IP address of the WiFi interface. Default 192.168.2.1
- ***Subnet Mask***: subnet mask of the WiFi interface
- ***DNS1***: Primary DNS server
- ***DNS2***: secondary DNS server
- ***IP Gateway***: allows you to specify the gateway when the local IP address is static in "WiFi Client" mode.

- ***Internet Access***: enable this feature to to access the Internet from a WiFi-enabled device.
- **LAN Access**: enable this feature to access devices connected to the device's Ethernet port from a WiFi-enabled device.



Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.3.2- WiFi → DHCP Server

This section allows you to enable and configure the DHCP server assigned to the WiFi interface of the TITAN-based device.

- ***Enabled***: checking the box will enable the DHCP server on the WiFi interface.
- ***Starting IP Address***: indicates the first IP address to be assigned by the DHCP server
- ***Ending IP Address***: indicates the last IP address to be assigned by the DHCP server
- ***MAC Address / IP Address****:* these two parameters allow the DHCP server to always assign the same IP address to a given MAC address.

Additional Notes.

- Once the configuration process is complete, click on the “SAVE CONFIG” button to save the changes. Press the “SAVE RULE” button for each MAC / IP pair you wish to create. You can create up to 10. Remember that you must restart the device for the new changes to take effect

4.4- Firewall

Section for configuring the device's security features.

4.4.1- Firewall >> NAT

Section from which ports can be mapped for external access to internal devices connected to the device. For example, if you have an IP camera connected to the Ethernet port of the TITAN-based device and you wish to access the camera from a remote computer, this section must be configured appropriately. You can create up to a total of 10 rules.

- ***Service name***: descriptive name of a mapping rule.
- ***Protocol*** indicates the port mapping protocol. TCP, UDP or both.
- ***Input port***: indicates the input port to the TITAN-based device via the WAN interface.
- ***Output port***: indicates the output port to the internal device connected to the TITAN-based device.
- ***Server IP Address:*** IP address of the device to be externally controlled (e.g. IP address of a camera).

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.
- You must also remember that in order to perform a NAT correctly:

  1.- The LAN IP address of the device to be controlled must be within the network range of the LAN IP address of the TITAN-based device.

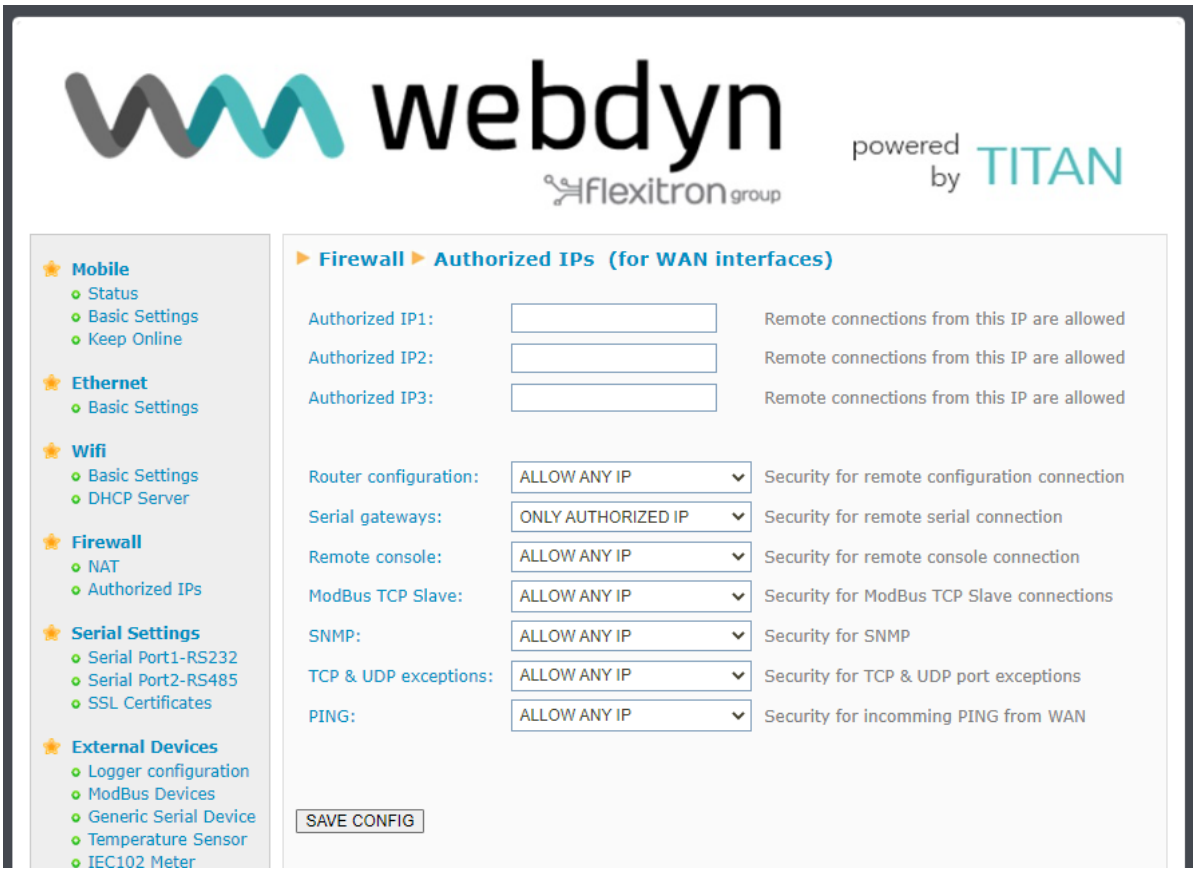
  2.- The Gateway IP address of the device to be controlled must be the LAN IP address of the TITAN-based device. See the examples in the Appendix for more information.

- It is possible to specify the interface for NAT. If you are unfamiliar with the feature, please use the “auto” option. This option, for example, could be useful for OpenVPN if you wish to access devices connected to the Ethernet port via NAT (and thus, which have the same configuration for remote access to different points).

4.4.2 - Firewall → Authorized IPs

This screen allows you to define up to 3 IP addresses authorised to accept connections on the WAN interface for the different services provided by the device. For example, if an authorised IP address of 90.166.108.200 is specified (such as the IP of an office, for example), certain services of the TITAN-based device will only be accessible from that IP address.

- ***Authorized IP1***: authorised IP address number 1
- ***Authorized IP2***: authorised IP address number 2
- ***Authorized IP3***: authorised IP address number 3
- ***Router configuration***: specifies whether remote connections to the web configuration environment are accepted from any IP, or only from authorised IP addresses.
- ***Serial Gateways***: specifies whether remote connections to 4G/3G/2G-RS232/485 gateway services are accepted from any IP, or only from authorised IP addresses.
- ***Remote console***: specifies whether to accept remote connections to the remote console service (telnet or SSH) from any IP, or only from authorised IP addresses.
- ***NAT***: specifies whether to accept remote connections to the device's mapped ports from any IP, or only from authorised IP addresses.
- ***Modbus TCP Slave****:* specifies whether remote connections to the Modbus TCP Slave service of the TITAN-based device are accepted from any IP, or only from authorised IP addresses.
- ***SNMP****:* specifies whether to accept remote connections to the SNMP service of the TITAN-based device from any IP, or only from authorised IP addresses.
- ***OpenVPN****:* specifies whether to accept remote connections to the "OpenVPN Server" service of the TITAN-based device from any IP, or only from authorised IP addresses.
- ***PING****:* will only respond to PING requests made from authorised IPs.
- ***Outgoing Connections****:* lets you specify whether the TITAN-based device can provide Internet access to all IP addresses, or only to authorised IP addresses. For example, imagine you only want Ethernet- or WiFi-enabled devices to be connected to the TITAN-based device to be able to send data to your server, preventing the device from being misused for other actions (Internet browsing, etc.)



Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.
- IP restrictions only apply at WAN connectivity level. WiFi access is protected by the WPA2 network key itself. That is, if you have WiFi and need security, activate WPA2 in the WiFi > Basic Settings section.
- If you use restrictions on “Outgoing connections”, remember that you also need to specify the IP address of the DNS server, so that it can be used in the event that domain names are employed rather than IP addresses.
- If you need more than 3 authorised IP addresses, you can specify more than one IP address in any box, separating them by “,”.

4.4.3 - Firewall → MAC Filter

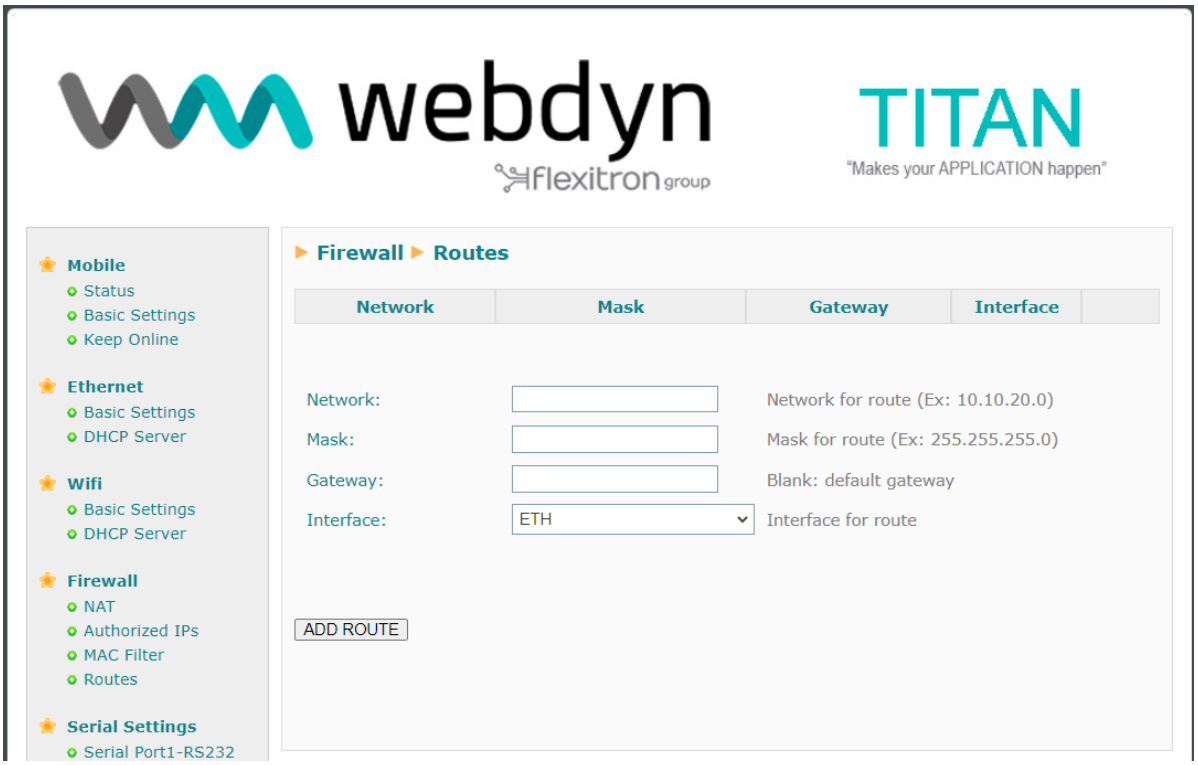
In this section, you can configure the device to filter communications over the Ethernet (or Wifi) interface based on MAC addresses.

- ***Mode*** - All MACs allowed, except those listed: All MAC addresses are allowed except the MAC addresses indicated in the list, which will be blocked. This is the default configuration.
- ***Mode*** - Only listed MACs allowed: All MAC addresses are blocked, except for the MAC addresses indicated in the list, which will be allowed.
- **MAC List**: List of allowed or blocked addresses, depending on the mode of operation.

4.4.4 - Firewall → Routes

In this section, you configure the device to add up to five routing paths per interface.

- ***Network*** - Indicates the network for the route to be configured.
- ***Mask*** - Indicates the netmask for the route to be configured.
- ***Gateway*** - IP address of the gateway for the route to be configured.
- ***Interface*** - Physical interface of the route to be configured.



4.5- Serial Settings → Serial portX

In the Serial Settings section it is possible configure each serial port of the TITAN-based device (speed, parity, etc.), as well as the special function that each of them will have (IP-serial gateways, Modbus TCP-RTU gateways, etc.).

4.5.1.- Serial Settings >> Serial portX

TITAN-based devices with multiple serial ports. These ports can be used to create 4G/3G/2G-Serial and CSD-Serial gateways to remotely control RS232 or RS485 devices. The number of RS232 or RS485 serial ports will depend on the TITAN-based device model used.

- ***Baudrate***: specifies the speed of the serial port (115200, 300)
- ***Data bits***: specifies the number of data bits (7, 8)
- ***Parity***: specifies the parity (none, even, odd)
- ***Stop bits***: number of stop bits (1, 2)
- ***Flow Control:*** specifies flow control (none or hardware)
- ***Timeout ms:*** indicates the number of milliseconds the device will wait without receiving data through the serial port before sending the data with IP. If you specify a "0" (default value), the data will be sent via IP as it arrives at the serial port. A value of 10, for example, specifies that no data is sent if a period of at least 10ms has not passed without receiving data at the serial port. This ensures that the data arrives at the destination less fragmented.

- ***Allow local embedded AT Commands***: checking this box enables the sending of LOCAL AT commands embedded in a Server or Client 4G/3G/2G-Serial gateway. AT commands must be sent through the RS232 or RS485 serial port of the TITAN-based device, but placed between the <MTXTUNNEL> and </MTXTUNNEL> tags. For example, if coverage is desired, the <MTXTUNNEL>AT+CSQ</ MTXTUNNEL> command can be sent. Or, if you wish to re-set a device remotely, you can send the command <MTXTUNNEL>AT^MTXTUNNEL=REBOOT </MTXTUNNEL>

- ***Allow remote embedded AT Commands***: checking this box enables the sending of REMOTE AT commands embedded in a Server or Client 4G/3G/2G-Serial gateway. AT commands must be sent over the established gateway connection but placed between the <MTXTUNNEL**R**> and </MTXTUNNEL**R**> tags. For example, if coverage is desired, the <MTXTUNNEL R>AT+CSQ</ MTXTUNNEL> command can be sent. Or, if you wish to re-set a device remotely, you can send the command <MTXTUNNEL**R**>AT^MTXTUNNEL=REBOOT </MTXTUNNEL**R**>

- ***Allow incoming GSM call*** **(CSD Data Call)***:* checking this box indicates that a CSD type call is accepted via the serial port. Only valid for operation when the port is configured as a TCP Server or TCP Client gateway, and, for some cases (such as for IEC-102 meter readings), when it is configured as "Nothing or Used by External Devices". When a CSD call is received and accepted, IP connections are suspended until the CSD call is terminated.

- ***Function: Nothing or Used by External Device***: select this operating option if you do not want to use a given serial port as an IP/serial gateway, or if you want the serial port (RS232 or RS485) to be used by an external device specified in the "External Devices" configuration section. For example, if you want to use a temperature sensor or any other device specified in the "External Devices" menu, you must select this option.

- ***Function: Serial – IP Gateway*** ***(TCP Server)***: select this operating option if you want to create a 4G/3G/2G TCP Server-Serial gateway, i.e.: a scenario in which the TITAN-based device is listening on a certain TCP port, waiting to receive a connection to establish the gateway.
- ***TCP Local Port***: TCP listening port for the 4G/3G/2G Serial Gateway
- ***Timeout****:* time (seconds) without data on the gateway to close the socket automatically.
- ***TCP Local Priority Port:*** priority TCP listening port for the 4G/3G/2G Serial Gateway. If there is an active connection in this port, connections within the TCP Local Port will not be allowed. Useful for reading electricity meters with IP connection priority.
- ***Close IEC102 session:*** select this option only if you are using the IP-Serial gateway for meter reading with IEC-60870-5-102 protocol. When this option is activated, the Titan router analyses the circulating packets of the IEC-60870-5-102 protocol, obtaining the link address and measurement point of the meter. This information is used each time an IP-Serial gateway is initiated against the meter in order to close the session with the meter that may be open.
- ***Temporary RS232 client****:* select this option if you wish to activate a temporary socket while you are working in TCP Server mode but the IP/Serial gateway is not established and you receive data through the serial port. If you activate this option, the parameters "Remote IP", "Remote TCP Port" and "ID String" of the TCP Client section below must be properly configured.
- ***Temporary client Wakeup****:* select this option if you wish to activate a temporary socket while you are working in TCP Server mode but the IP/Serial gateway is not established and the configured time has been reached. If you activate this option, the parameters "Remote IP", "Remote TCP Port" and "ID String" of the TCP Client section below must be properly configured.
- ***Temporal client Time****:* indicates the time a temporary socket is active.
- ***Temporary client Random***: it is possible to set a random time in seconds for a temporary client to be added to the Wakeup time. This is only useful if you have a large number of

devices connecting at the same time and you wish to enter a random time to distribute the number of simultaneous connections to a server.

- ***SSL/TLS enabled**:* if TCP Server gateways need to be secured, you can do so using this option. If activated, the certificate section "Serial Settings → SSL Certs" must also be configured.

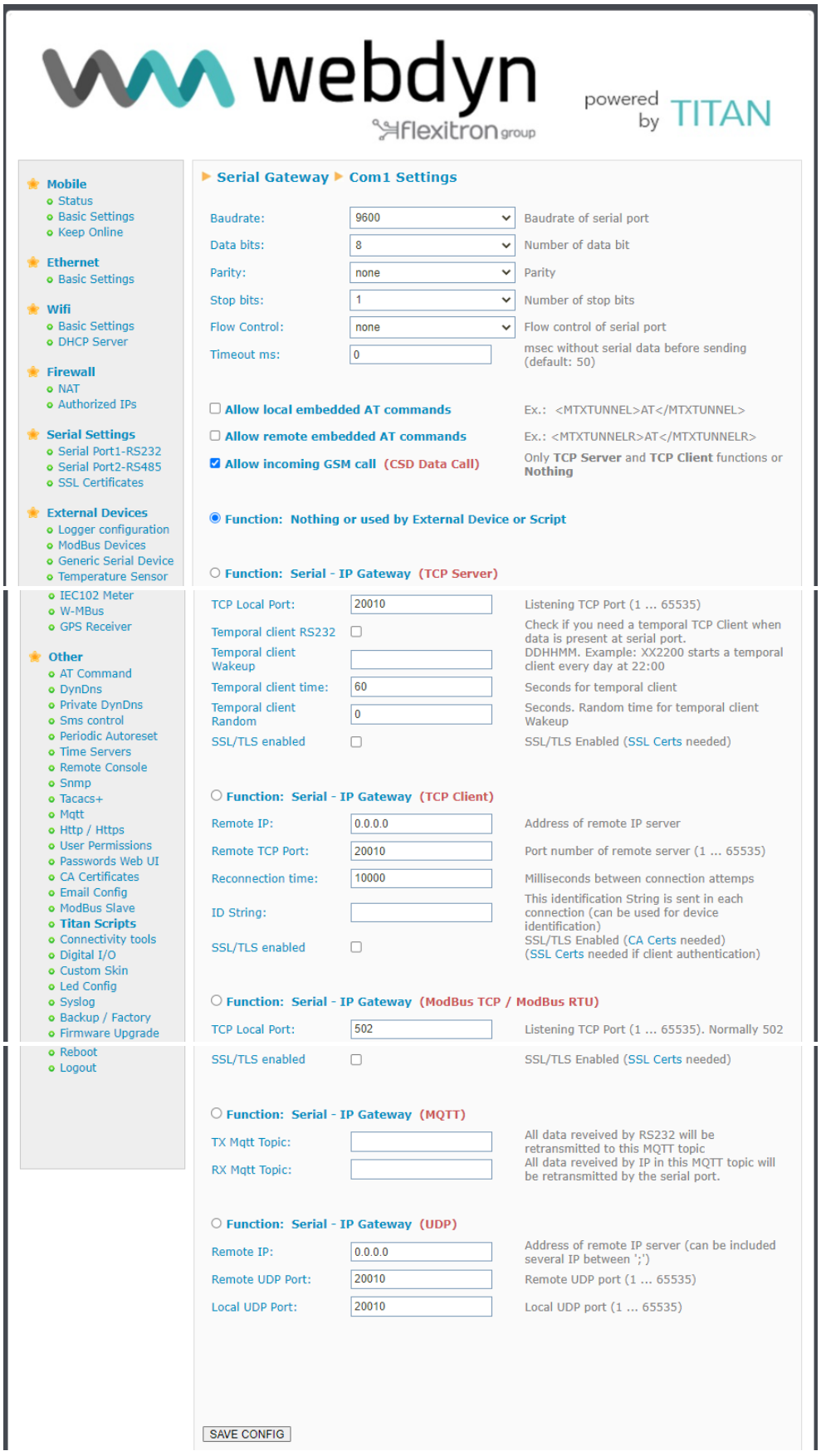
- ***Function:*** Serial - IP Gateway ***(TCP Client)***: select this operating option if you wish to establish a transparent Serial - 4G/3G/2G gateway in TCP Client mode, i.e.: a scenario where the TITAN-based device connects to a given IP / TCP port to establish the Serial - 4G/3G/2G gateway
- ***Remote IP***: the IP address to which the TITAN-based device will be connected
- ***Remote TCP Port***: the TCP port to which the TITAN-based device will be connected
- ***Reconnection time***: in case of connection failure or connection problems, this indicates how many milliseconds should be left between connection attempts. 0 = immediate reconnection. Be careful with this value if you do not have a flat rate SIM card or if you have a very low contracted monthly data volume.
- ***ID String***: string that is sent just after connecting the socket to the remote IP. This text will allow you to identify the device making the connection. For example, if you have 100 TITAN-based devices in this operating mode, you will be able to determine which of these 100 devices has made a particular connection. It is possible to add the special tags [IMEI] , [CR] and [LF] which will be substituted by the IMEI, and the characters 0x13 and 0x10 respectively.

- ***Function:*** Serial - IP Gateway ***(Modbus TCP / Modbus RTU)***: select this operating option if you wish to establish a Serial - 4G/3G/2G gateway with Modbus TCP - Modbus RTU protocol conversion. Do not use this option if your control software uses Modbus RTU; a normal TCP Server gateway would suffice. Use this option if your control software uses a Modbus TCP protocol. If you have any questions, please contact@webdyn.com

- ***Serial - IP Gateway*** ***(MQTT)***: select this option if you wish to create an IP-Serial gateway via a MQTT protocol. You must configure the MQTT client to use this option, which you will find under "Other - MQTT".
- ***TX Mqtt Topic***: the TITAN-based device will resend, to the MQTT topic defined in this parameter, all the data received through its serial port (data sent in RAW mode).
- ***RX Mqtt Topic***: the TITAN-based device will subscribe to the MQTT topic defined in this parameter and will resend all the data received in this topic (data sent in RAW mode) through its serial port.

- ***Function: Serial – IP Gateway*** ***(UDP)***: select this operating option if you wish to create a transparent serial - IP gateway via a UDP protocol.
- ***Remote IP***: IP address of the remote server where the data will be sent via UDP

- ***Remote UDP Port***: UDP port of the remote server
- ***Local UDP Port***: the device's listening UDP port


webdyn
Flexitron group
powered by TITAN
Mobile
Status
Basic Settings
Keep Online
Ethernet
Basic Settings
Wifi
Basic Settings
DHCP Server
Firewall
NAT
Authorized IPs
Serial Settings
Serial Port1-RS232
Serial Port2-RS485
SSL Certificates
External Devices
Logger configuration
ModBus Devices
Generic Serial Device
Temperature Sensor
IEC102 Meter
W-MBus
GPS Receiver
Other
AT Command
DynDns
Private DynDns
Sms control
Periodic Autoreset
Time Servers
Remote Console
Snmp
Tacacs+
Mqtt
Http / Https
User Permissions
Passwords Web UI
CA Certificates
Email Config
ModBus Slave
Titan Scripts
Connectivity tools
Digital I/O
Custom Skin
Led Config
Syslog
Backup / Factory
Firmware Upgrade
Reboot
Logout
Serial Gateway ▸ Com1 Settings
Baudrate: 9600 Baudrate of serial port
Data bits: 8 Number of data bit
Parity: none Parity
Stop bits: 1 Number of stop bits
Flow Control: none Flow control of serial port
Timeout ms: 0 msec without serial data before sending (default: 50)
Allow local embedded AT commands Ex.: <MTXTUNNEL>AT</MTXTUNNEL>
Allow remote embedded AT commands Ex.: <MTXTUNNELR>AT</MTXTUNNELR>
Allow incoming GSM call (CSD Data Call) Only TCP Server and TCP Client functions or Nothing
Function: Nothing or used by External Device or Script
Function: Serial - IP Gateway (TCP Server)
TCP Local Port: 20010 Listening TCP Port (1 ... 65535)
Temporal client RS232 Check if you need a temporal TCP Client when data is present at serial port.
Temporal client Wakeup DDHHMM. Example: XX2200 starts a temporal client every day at 22:00
Temporal client time: 60 Seconds for temporal client
Temporal client Random 0 Seconds. Random time for temporal client Wakeup
SSL/TLS enabled SSL/TLS Enabled (SSL Certs needed)
Function: Serial - IP Gateway (TCP Client)
Remote IP: 0.0.0.0 Address of remote IP server
Remote TCP Port: 20010 Port number of remote server (1 ... 65535)
Reconnection time: 10000 Milliseconds between connection attemps
ID String: This identification String is sent in each connection (can be used for device identification)
SSL/TLS enabled SSL/TLS Enabled (CA Certs needed)
(SSL Certs needed if client authentication)
Function: Serial - IP Gateway (ModBus TCP / ModBus RTU)
TCP Local Port: 502 Listening TCP Port (1 ... 65535). Normally 502
SSL/TLS enabled SSL/TLS Enabled (SSL Certs needed)
Function: Serial - IP Gateway (MQTT)
TX Mqtt Topic: All data reveived by RS232 will be retransmitted to this MQTT topic
RX Mqtt Topic: All data reveived by IP in this MQTT topic will be retransmitted by the serial port.
Function: Serial - IP Gateway (UDP)
Remote IP: 0.0.0.0 Address of remote IP server (can be included several IP between ';')
Remote UDP Port: 20010 Remote UDP port (1 ... 65535)
Local UDP Port: 20010 Local UDP port (1 ... 65535)
SAVE CONFIG

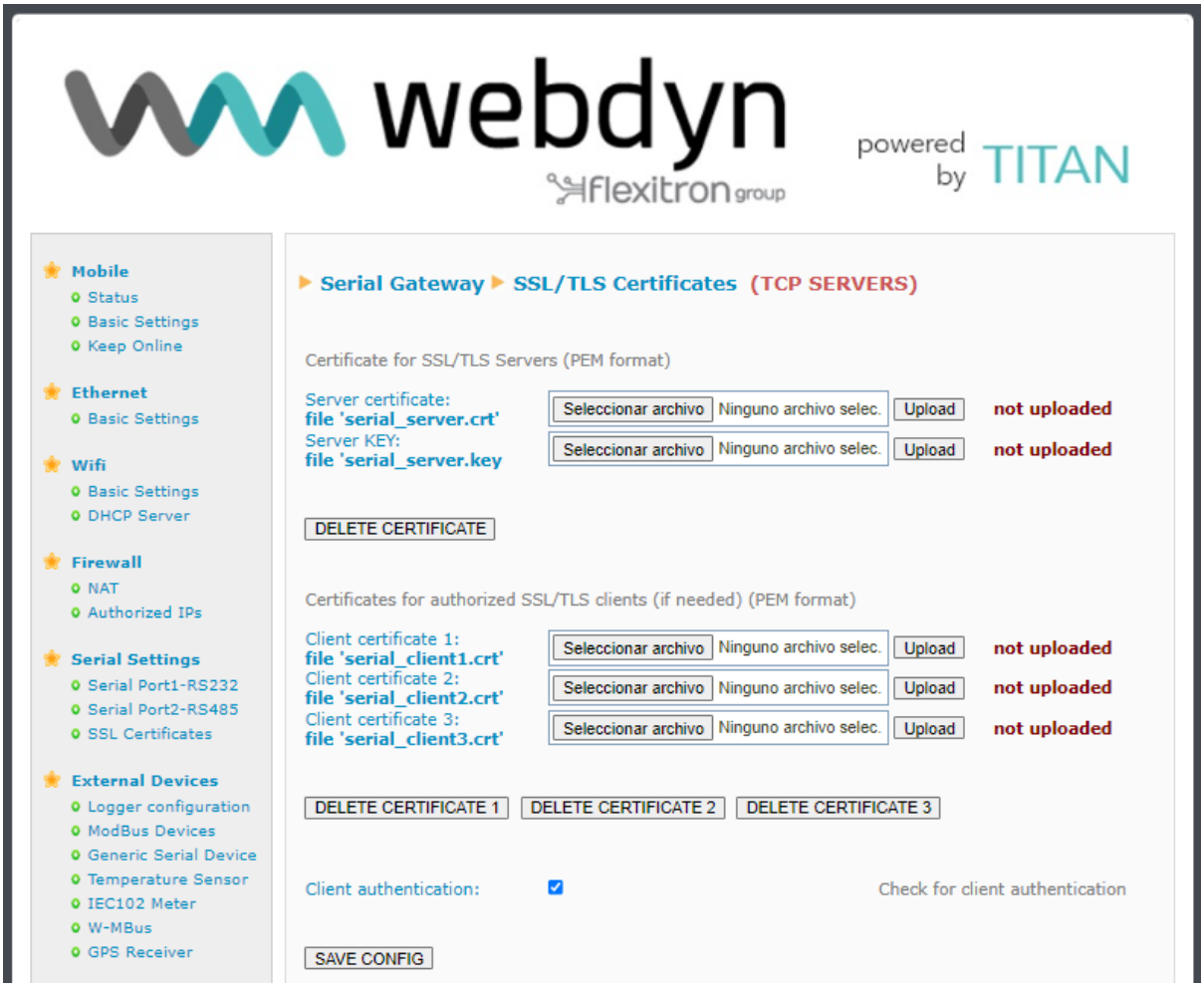
Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.
- If a TTL serial port appears, it will correspond to an internal serial port of the TITAN-based device for special communication cards (RF868 MHz, GPS, etc.), i.e.: it is not an external serial port of the device.

4.5.2.- Serial Settings → SSL Certificates

The IP-Serial gateways of the TITAN-based device can be established using SSL/TLS connections, both on TCP client and TCP server connections. The corresponding certificate section will need to be configured for SSL/ TLS connections.

The **Serial Gateway SSL/TLS → Certificates (TCP SERVERS)** section refers to certificates when the COM port has been configured as a TCP Server socket.



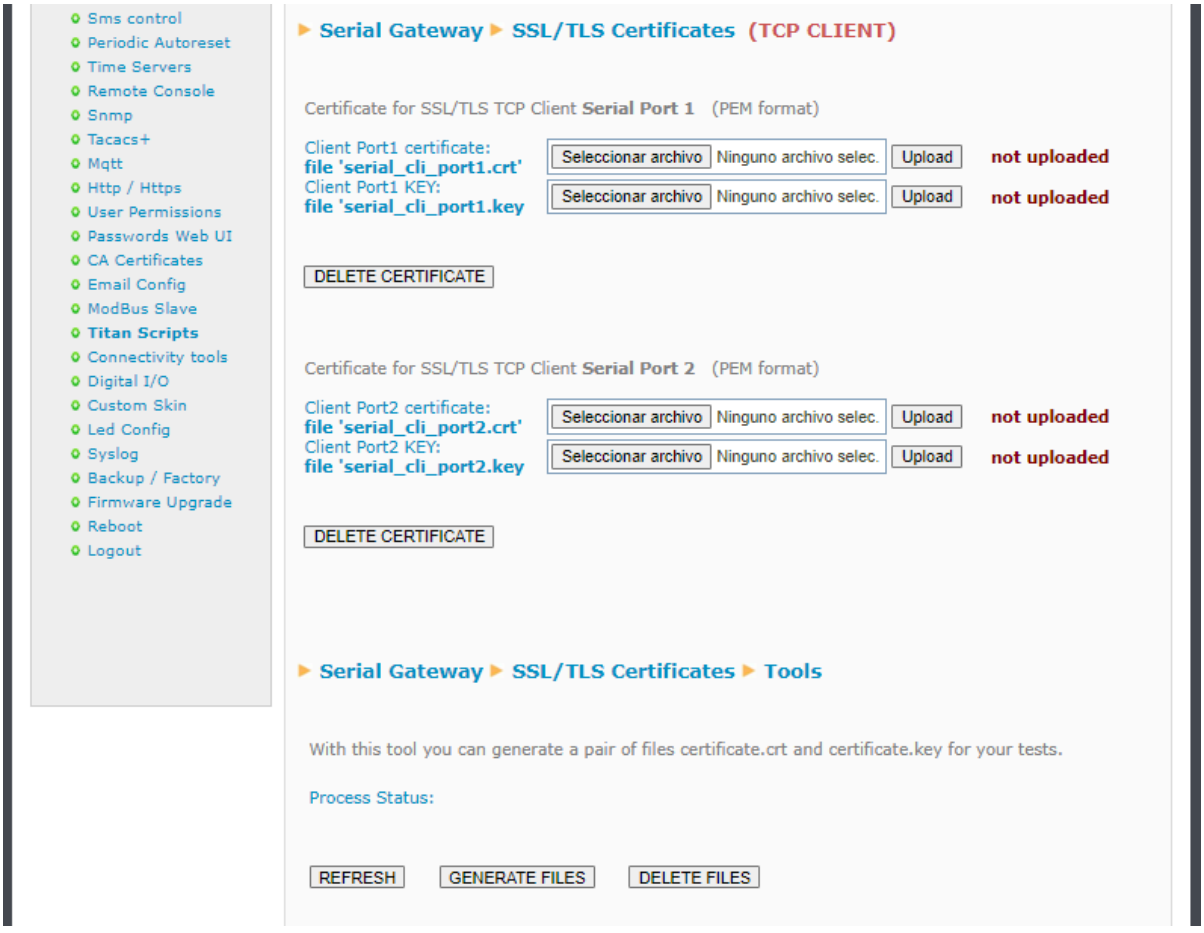
- ***Server Certificate:*** certificate (PEM format) to be used by the TITAN-based device for any service configured as a TCP Server.
- ***Server KEY:*** private key of the certificate used by the TITAN-based device when configured by a TCP Server.

- ***Client authentication:*** if this box is checked, the TITAN-based device will only allow clients using the authorised client certificates to connect to its TCP Server sockets.

- ***Client Certificate 1:*** this field allows you to enter the authorised certificate 1 in the event "Client authentication" is used.
- ***Client Certificate 2:*** this field allows you to enter the authorised certificate 2 in the event "Client authentication" is used.
- ***Client Certificate 3:*** this field allows you to enter the authorised certificate 3 in the event "Client authentication" is used.

The **Serial Gateway → SSL/TLS Certificates (TCP CLIENT)** section refers to certificates when the COM port has been configured as a TCP Client socket.

Certificates for TCP Client sockets are only required when the remote TCP server (where the TITAN-based machine will connect) requires authentication by certificate.



- ***Client Port1 Certificate:*** certificate (PEM format) to be used by TITAN-based devices when a Port1 serial port acts as a TCP client with SSL/TLS and certificate identification.
- ***Client Port1 KEY:*** private key of the certificate used by the TITAN-based device when a Port1 serial port acts as a TCP client with SSL/TLS and certificate identification.

- ***Client Port2 Certificate:*** certificate (PEM format) to be used by TITAN-based devices when a Port2 serial port acts as a TCP client with SSL/TLS and certificate identification.
- ***Client Port2 KEY:*** private key of the certificate used by the TITAN-based device when a Port2 serial port acts as a TCP client with SSL/TLS and certificate identification.

The **Serial Gateway → SSL/TLS Certificates →** section is a feature that allows you to easily generate a pair of .crt and .key files for performance testing.

4.6- External Devices

The internal datalogger is configured in this section, as well as external serial peripherals (temperature probe, Modbus RTU or Modbus TCP devices, generic RS232-485 devices, etc.).

4.6.1- External Devices >> Logger configuration

If you need the TITAN-based device to collect data from external devices (Modbus devices, temperature sensors, generic serial devices, GPS receivers, W-Mbus devices, IEC-102 meters, etc.) and then send them to an external server, you first need to configure the internal Logger, that is, how to collect the data, and where and how to send them. That is what this section is for.

This section allows you to configure the parameters related to the internal datalogger. You can send the data to a server via HTTP/HTTPS, FTP/FTPS or MQTT/MQTTS. In all cases, the data will be sent in JSON format

General parameters:

- ***ID***: Optional. This is a text parameter that allows you to enter an identifying string included within the JSON file.
- ***Send mode****:* FIFO or LIFO. In FIFO mode, data is transmitted to the server in order of arrival, i.e.: the data that enters the datalogger first is the first to be sent. LIFO mode operates the other way around, with the last data to be received being sent first. LIFO mode can be useful in applications where you always want to display data on screen in real time. For example, a device may be without coverage for several days, storing large amounts of data without being

able to transmit them. Once coverage returns, it may take some time for all accumulated data to be sent to the server. In LIFO mode, by sending the last data to be collected first, real-time data can be quickly displayed on a screen while the rest of the previously captured data arrives later.

- ***Time format***: the JSON data object always includes a timestamp (date and time) with the time at which the stored data was recorded. This parameter allows you to specify the format of the timestamp.
- **Use script**: if this box is checked, before the TITAN-based device stores collected data (e.g.: a Modbus reading) in its internal memory (datalogger) in the default JSON format of the TITAN-based device, the data can be formatted to be stored in another format. This can be very useful for sending data to platforms that require a specific structure. See the application note related to the 'JSON Transformer Script' for more information.

  For example, Modbus registers read by TITAN-based devices would be stored as standard in the following format:

  {"IMEI":"869101054287806","TYPE":"MODB","TS":"2022-06-03T11:38:01Z","ID":"TRES","A":"192.168.1.28:502","ST":"1","N":"6","V":[1,2,3,4,5,6],"P":"ID0001"}

  But you may wish to change this structure, add new custom fields, etc.

  {"data":{"IMEI":"869101054287806","TYPE":"MODB","TS":"2022-06-03T11:38:01Z","ID":"TRES","A":"192.168.1.28:502","ST":"1","N":"6","V":[1,2,3,4,5,6],"P":"ID0001"**,"myField1":123,"MyField2":456**}**}**

  This format change can be done through "JSON Transformer Scripts"

- **Use array**: by default, TITAN-based devices send data to a server (MQTT, HTTP) on a register-by-register basis, i.e.: one transmission per register sent. If you select the "Use array" option, the data is sent as an array, allowing up to 100 readings to be sent simultaneously and greatly speeding up the data-sending process.

  For example, if this box is not checked, the data is transmitted in the standard JSON format of TITAN-based devices

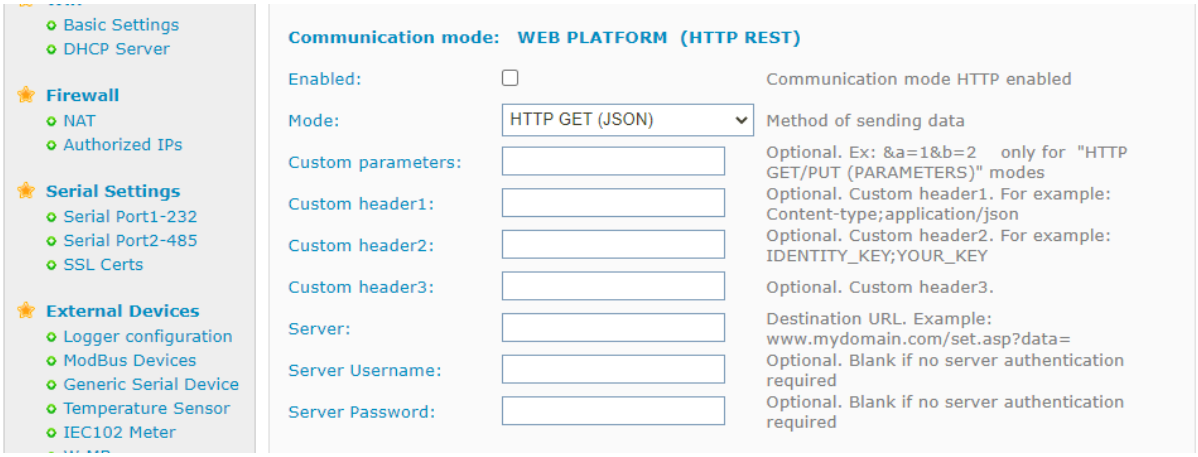
  {"IMEI":"869101054287806","TYPE":"MODB","TS":"2022-06-03T11:38:01Z","ID":"TRES","A":"192.168.1.28:502","ST":"1","N":"6","V":[1,2,3,4,5,6],"P":"ID0001"}

  If the box is checked, the data is transmitted as an array with content containing between 1 and 100 items, depending on the data stored in the datalogger.

  [{"IMEI":"869101054287806","TYPE":"MODB","TS":"2022-06-03T11:38:01Z","ID":"TRES","A":"192.168.1.28:502","ST":"1","N":"6","V":[1,2,3,4,5,6],"P":"ID0001"}**,**{"IMEI":"869101054287806","TYPE":"MODB","TS":"2022-06-

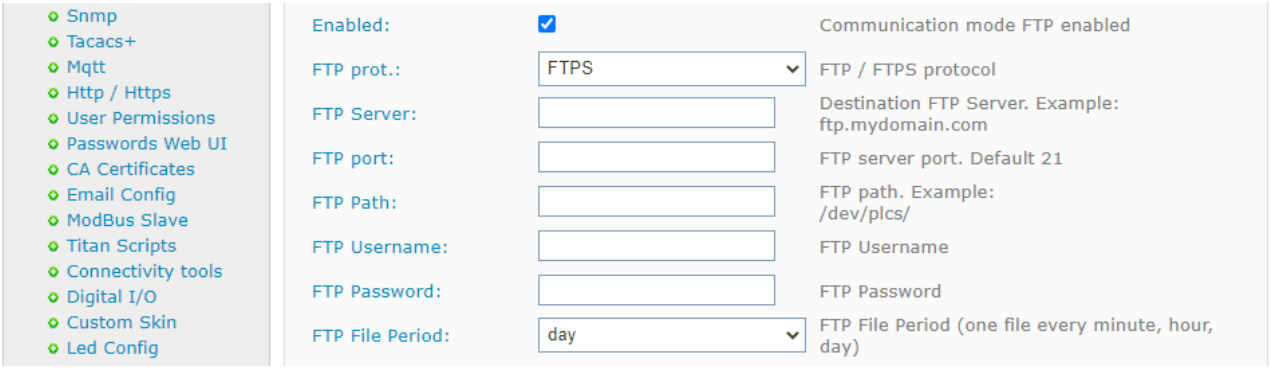
03T11:39:01Z","ID":"TRES","A":"192.168.1.28:502","ST":"1","N":"6","V":[1,2,3,4,5,6],"P":"ID0001"},{"IMEI":"869101054287806","TYPE":"MODB","TS":"2022-06-03T11:40:01Z","ID":"TRES","A":"192.168.1.28:502","ST":"1","N":"6","V":[1,2,3,4,5,6],"P":"ID0001"}, ...]

HTTP mode



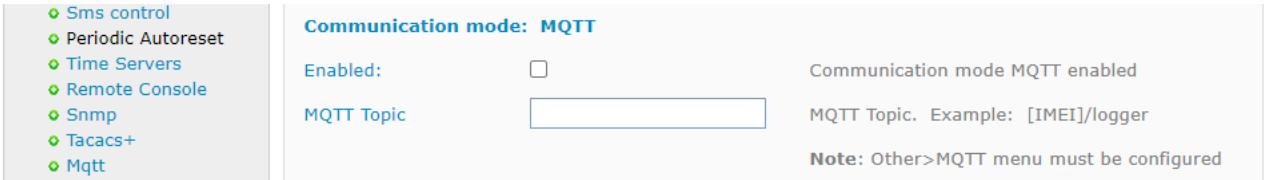
- ***Enabled***: click to enable the mode for sending data to a web platform via HTTP/HTTPs.
- ***Mode***: data-sending mode. You can choose between HTTP GET (JSON), HTTPS GET (JSON), HTTP POST (JSON) and HTTPS POST (JSON)
- ***Custom parameters***: allows you to add optional parameters for HTTP GET methods.
- ***Custom header1, Custom header2 and Custom header3***: allows you to add headers to HTTP/HTTPS requests. Many WEB platforms require a header with an identifying token. You can configure it in this section.
- ***Server***: Complete URL to send data collected in the datalogger. For example www.metering.es/json/set.asp?data=
- ***Server Login***: if your platform has restricted access, please enter your username here.
- ***Server Password***: if your platform has restricted access, please enter your user password here.

FTP mode



- ***Enabled***: click to enable the data-sending mode on an FTP server (the data file is in JSON format).
- ***FTP prot:*** allows you to select the sending protocol, with a choice between FTP or the secure FTPS protocol.
- ***FTP Server***: FTP server to send data (IP or DNS).
- ***FTP port***: FTP server port, usually 21.
- ***FTP Path***: path within the server where the data dump occurs.
- ***FTP Username***: the username of an account on your writeable FTP server.
- ***FTP Password***: the password of an account on your writeable FTP server.
- **FTP File Period**: how often you want TITAN to send the accumulated data file to your server (every day, every hour or every minute)

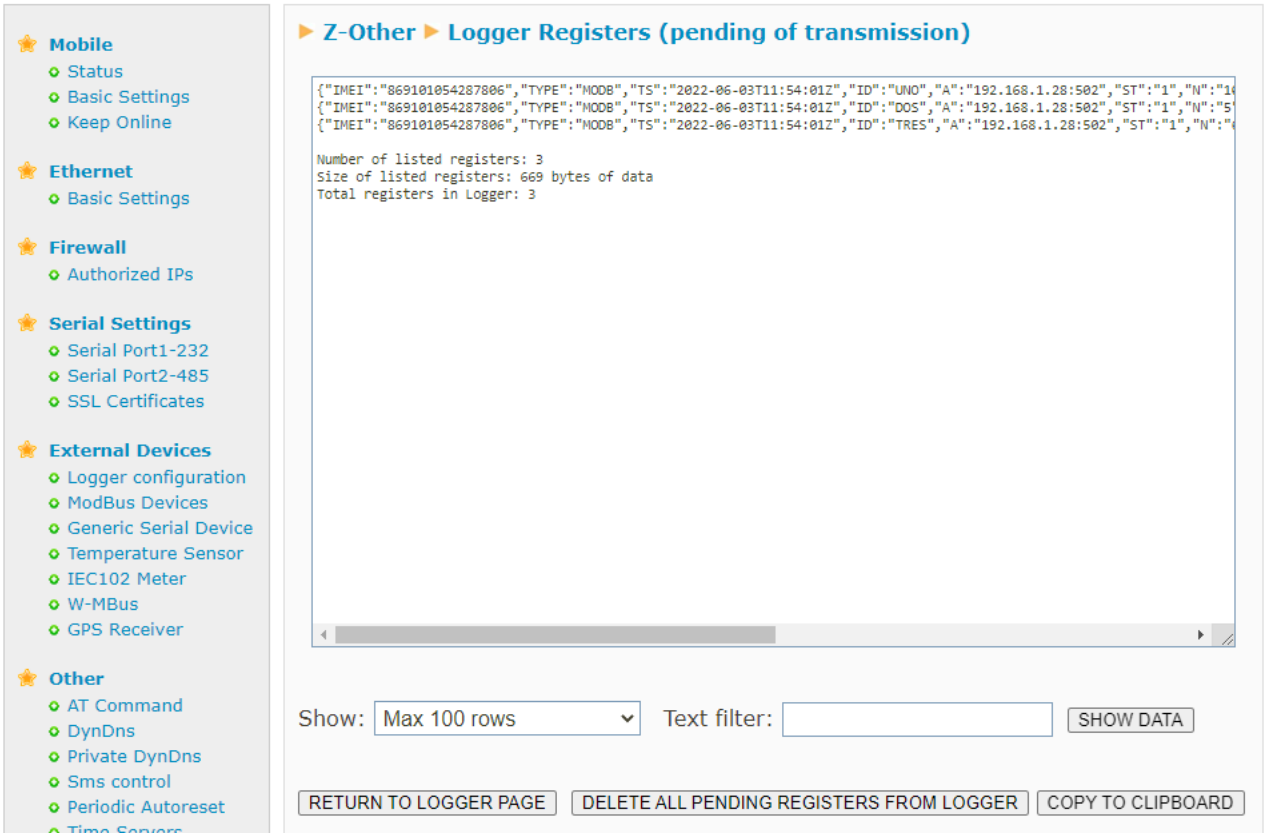
MQTT mode



- ***Enabled***: click to enable the data-sending mode on an MQTT broker (all data is sent in JSON format).
- ***MQTT Topic***: MQTT topic to which the stored logger data will be sent.

Real Time Status.

Allows you to view the status of the TITAN-based device's internal datalogger and to view the data pending transmission. It may also be useful for some applications where data transmission to a remote server is not desired.



In this section, you can also delete all the data pending transmission by the TITAN-based device by clicking on the "DELETE ALL PENDING REGISTERS FROM LOGGER" button. You can also filter by text.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.
- When using HTTP or MQTT send mode, data is sent to your server as it is collected. In the event of a lack of 4G/3G/2G coverage or problems sending data (e.g.: a remote

server crash), the TITAN-based device will store the data internally to be sent later when the communication is back to normal.

- When using FTP send mode, the data is stored in a JSON register structure file and sent according to the time interval selected for sending (minute, hour or day). In the event of a lack of GSM coverage or problems sending data (e.g.: a remote server crash), the TITAN-based device will store the data internally to be sent later when the communication is back to normal.
- The name of the FTP file created on your server will take the following format:

  IMEI-year-month-day.txt → If the selected data-sending period is "every day".

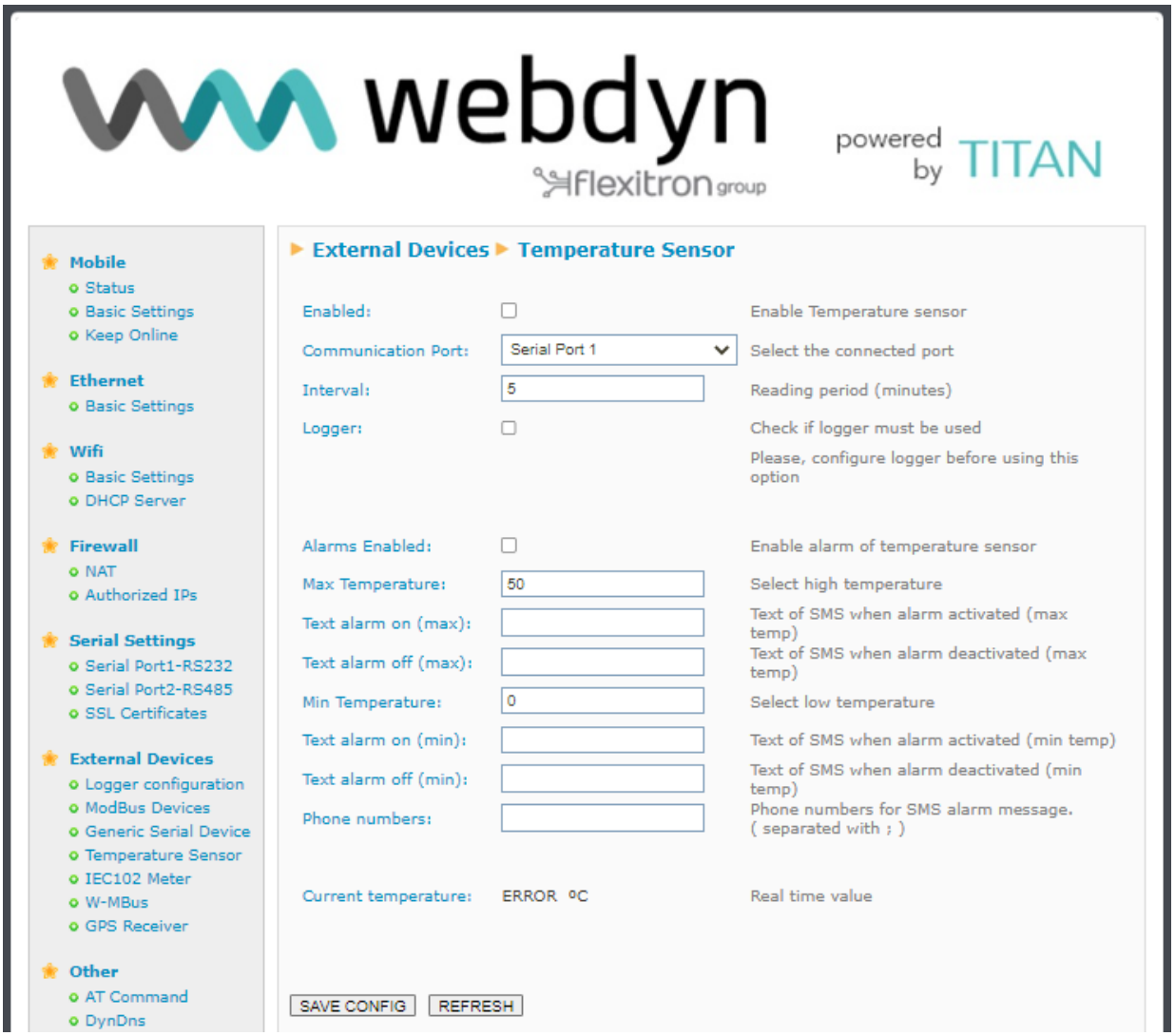
  IMEI-year-month-day-hour.txt → If the selected data-sending period is "every hour".

  IMEI-year-month-day-hour-minute.txt → If the selected data-sending period is "every minute".
- If you use the MQTT mode, remember to properly configure the "OTHER → Mqtt" section. In this section, you must enter all the configuration parameters necessary to connect the TITAN-based device to your MQTT broker

4.6.2 - External Devices → Temperature Sensor

If your TITAN-based device model supports a MTX-Temp-RS232 temperature probe, you can send the temperature periodically to a web platform, send an SMS alert message when the temperature is above or below a certain threshold, etc.

- ***Enabled***: Check this box if you have an MTX-Temp-RS232 temperature probe connected to a serial port.
- ***Serial Port***: Select the serial port of the device where the MTX-Temp-RS232 temperature probe is connected.
- ***Interval***: If you intend to send periodic temperature readings to your web server, please indicate the period here (in minutes).
- ***Logger***: If you wish to use the internal logger to store temperature data (to send to a web platform later), you must select this option. When should this option not be activated? For example, when you only want to send an SMS alert if the temperature is outside the range.
- ***Alarms enabled***: Select this option if you wish to activate the alarms for out-of-range temperatures.
- ***Max Temperature***: Maximum temperature above which a high temperature alarm is activated.
- ***Text alarm on (max)*:** Alarm text sent when a high temperature alarm is activated.
- ***Text alarm off (max)***: Alarm text sent when a high temperature alarm is deactivated.
- ***Min Temperature***: Minimum temperature below which a low temperature alarm is activated.
- ***Text alarm on (min)***: Alarm text sent when a low temperature alarm is activated.
- ***Text alarm off (min)***: Alarm text sent when a low temperature alarm is deactivated.
- ***Phone numbers:*** Telephone numbers (separated by a semicolon ";") to which to send the SMS alarm messages
- ***Current temperature***: indicates the current temperature in real time if an MTX-Temp-RS232 temperature sensor is connected.



Additional Notes.

- Once the configuration process is finished, click on the "SAVE CONFIG" button to save the changes. Remember that you must restart the device for the new changes to take effect.
- The hysteresis value used by the device is 1 degree. For example, if you set the high temperature to 50ºC, the alarm will go off when the temperature reaches 50ºC, but the alarm will not be deemed to have stopped until it drops to at least 49º. This prevents constant SMS messages being sent when the temperature is just at the selected threshold.
- Example of the JSON format frame sent:

{"IMEI":"358884050088207","TS":"25/12/2014 17:40:09","TYPE":"TEMP", "P":"1234","TEM":25.5,"TEMH":0,"TEML":0}

Where:

IMEI: device identification number. Unique for each device

TS: timestamp DD:MM:YYYYY HH:MM:SS

TYPE: type of frame. In this case, temperature

P: Logger ID field (External Devices > Logger configuration)

TEM: temperature

TEMH: high temperature alarm (0 = no, 1 = yes)

TEML: low temperature alarm (0 = no, 1 = yes)

4.6.3 - External Devices → Modbus Devices

From firmware version 5.3.6.25 it is possible to choose between "Basic Mode" (the only mode existing until firmware 5.3.6.24) and "Expert Mode".

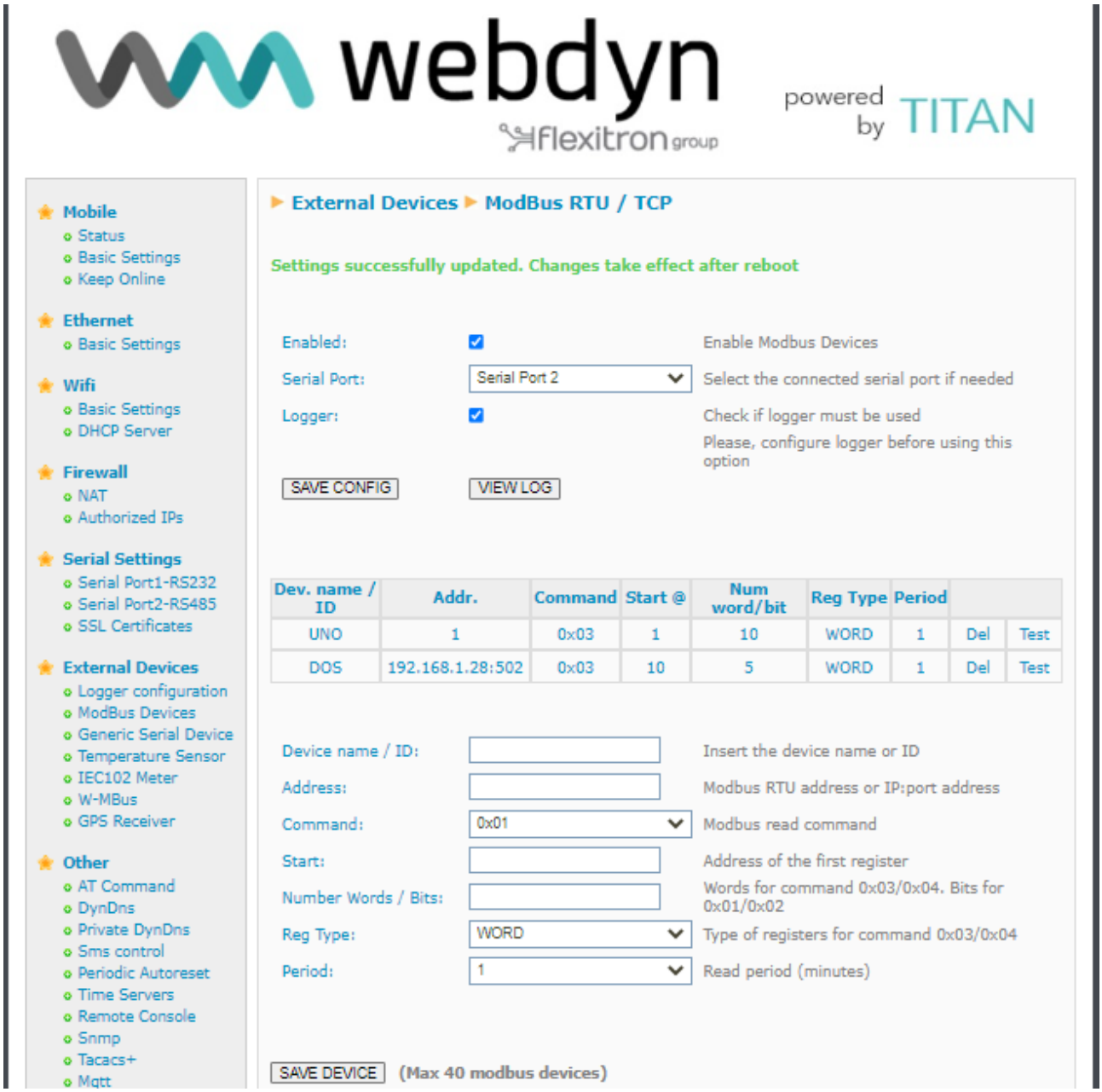
4.6.3.1 – Basic Mode

TITAN-based devices are equipped to read, store and send registers of external Modbus RTU and Modbus TCP devices to an external server (via HTTP/HTTPS, FTP or MQTT/MQTTS). That is, they can schedule a periodic reading of up to 40 Modbus RTU and/or TCP devices, selecting the reading registers, and then sending the readings to a server via a JSON object.

- ***Enabled***: Check this box if you have one or more Modbus RTU devices connected to a serial port and you intend to read Modbus registers autonomously.
- ***Serial Port***: Allows you to select the RS232 or RS485 serial port of the TITAN-based device where the Modbus RTU device is connected.
- ***Logger***: If you wish to use the internal logger to store the Modbus registers read (to send to a web platform later), you must select this option.

- ***Device name***: identification name of a Modbus RTU or Modbus TCP device.
- ***Address***: Modbus RTU address of the device to be read. If it is a Modbus TCP device, The IP and port (IP format: port) must be specified
- ***Command***: Modbus reading command.
- ***Start***: initial reading register
- ***Num Words***: number of registers to be read
- ***Reg type***: type of register to be read
- ***Period***: reading period, i.e.: after how many minutes the set of registers should be read each time.

Important: See application note AN27 (ANV6_27-Router-TITAN-Modbus-RTU-RTU-TCP-Concentrator- multimap_HTTP.pdf and ANV6_32-Router-TITAN-Modbus-RTU-TCP-Concentrator-multimap_MQTT.pdf) for detailed information and examples of how to use this functionality of TITAN-based devices.



Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.
- If you wish to create a new Modbus RTU device, you must fill in the form and press the "SAVE DEVICE" button.
- Here is an example of the JSON frame format stored with the readings to be sent to a server:

{"IMEI":"354740050367237”,"TS":"17/02/2014 19:02:46",”TYPE”:”MODB”, "P":"1234","ST":1,"A":1,"V":[1,2,3,4,5,6,7,8,9,10]}

Where:

IMEI: device identification number. Unique for each device

TS: timestamp DD:MM:YYYYY HH:MM:SS

TYPE: type of frame. In this case, Modbus

P: Logger ID field (External Devices > Logger configuration)

ST: initial register

A: Modbus address of device read

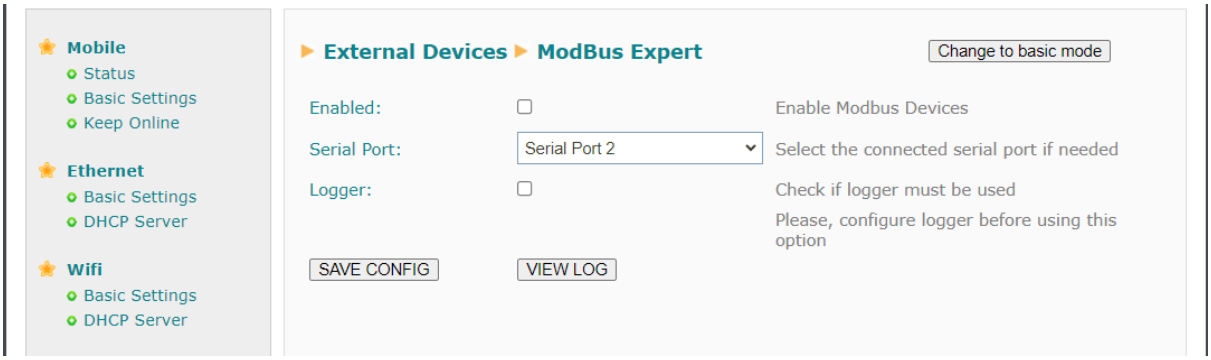
V: array containing the read data

- We recommend you read the chapter on available AT commands, because you can read and change values of Modbus registers via AT commands, in the web configuration environment, from Remote Console (Telnet), SSH or SMS, etc.

- **In the "Start" and "Num Words" fields, it is possible to create non-consecutive registers with firmware version 3.26 or more recent versions. See Application Note AN27 (ANV6_27-Router-TITAN-Modbus-RTU- TCP-Concentrator-multimap_HTTP.pdf) and AN32 ANV6_32-Router-TITAN-Modbus-RTU-TCP-Concentrator-multimap_MQTT.pdf) for further information and examples.**

4.6.3.2 – Expert Mode

"Expert Mode" is very similar to "Basic Mode", but with a significantly increased performance compared to its predecessor. The main display is divided into three different sections. The first is displayed as follows:

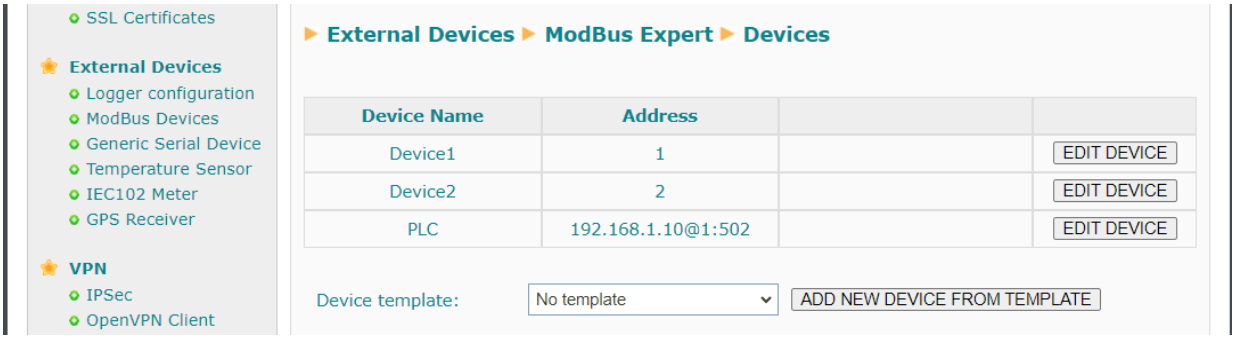


- ***Enabled***: Check this box to enable the Modbus “Expert Mode” service. If the Modbus "Basic mode" service is enabled, it will be disabled automatically, as both services cannot be used simultaneously.
- ***Serial Port***: Allows you to select the RS232 or RS485 serial port of the TITAN-based device where the Modbus RTU device is connected. You can also select the "None" option if you are only going to work with Modbus TCP devices. If you are going to work simultaneously with Modbus RTU and Modbus TCP devices, specify the serial port to which your Modbus RTU devices are connected ("Serial Port 1" for RS232 and "Serial Port 2" for RS485).
- ***Logger***: Check this box if you wish to store the Modbus register readings in the internal datalogger of the Titan router, ready to be sent at a later time to a WEB platform (HTTP, MQTT or FTP).

The following section presents the device templates, which are useful for adding multiple devices of the same type to an installation. By creating a template, you can quickly add a new template-based Modbus device without having to re-create the entire Modbus register map.

- ***Edit template button***: Allows you to edit the template indicated in the "Template" drop-down menu.
- ***Add new template***: Allows you to create a new template.
- ***Export CSV***: Allows you to export a template in CSV format, which is useful for importing it into other devices.

A table listing the Modbus devices created can be found in the final section of this display.

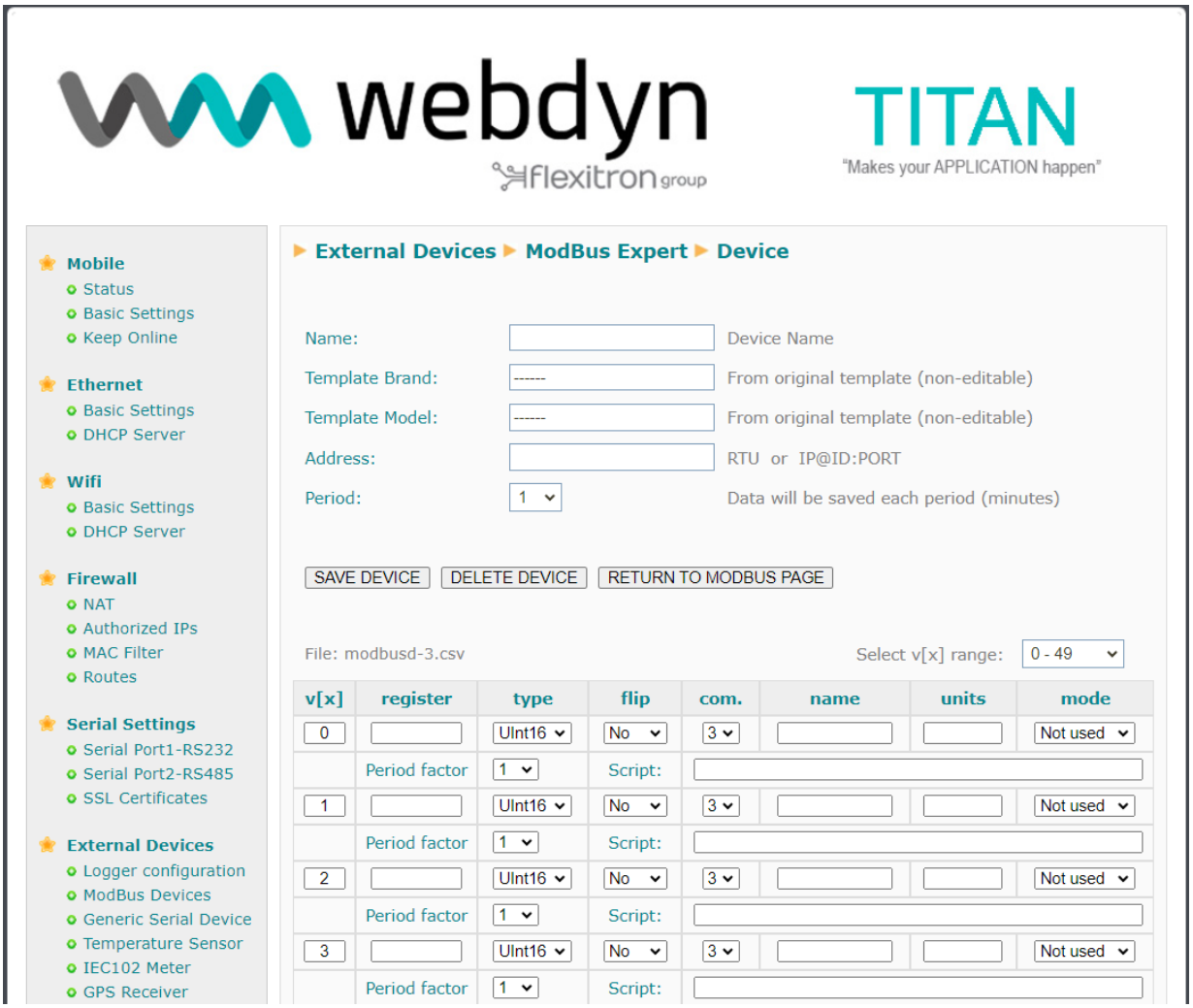


This table shows the name of the device and whether it has a Modbus RTU or Modbus TCP address. If you click on the "**EDIT DEVICE**" button of one of the devices, you will be directed to the editing screen for the corresponding Modbus registers of that device.

To add a new device to the list, simply choose a template from which you would like to start creating such device. If you do not wish to choose a template, simply select "No template" and click on the "**ADD NEW DEVICE FROM TEMPLATE**" button.

This final section features the "**RESTART MODBUS SERVICE**" button. If the Modbus Expert service is running, it will be possible to reload the configuration of all devices and restart the service. This will allow changes to be made to Modbus device configurations (modifications, deletion of devices and registration of new devices) and restart the service without needing to reboot the Titan router.

Device editing screen:



The upper section of the above display allows you to configure the following parameters:

- ***Name***: Device name
- ***Template Brand***: Uneditable — imported from the template used.

- ***Template Model***: Uneditable — imported from the template used.
- Address: Device Modbus RTU address or IP address in the format IP@ID:PORT, where IP is the IP address of the modbus TCP device, ID (optional) is its internal address (normally 1) and PORT is the TCP port. Valid examples: 192.168.1.100:502 , 192.168.1.100@1:502 It is also possible to set the address to "0" if you wish to disable the device at some point (not to be read) in the event that you need to remove it.
- Period: Indicates the base period, in minutes, in which the Modbus registers read will be stored inside the Titan router's datalogger, ready to be sent at a later time to a WEB platform. For example, if you specify 5, the registers read will be stored every 5 minutes by default.

The lower section shows the complete Modbus register map of the device:

- ***V[0]***: An array where the value read by the register is stored. This is important for the "Script" section, as explained later.
- ***Register***: The register number to be read must be specified, for example 40000. If the register is of the String variety, the register address must be indicated, followed by "_" and ending with the number of characters. For example, for a String of length 10 and starting at register 42000, "42000_10" should be specified.
- ***TYPE:*** The type of Modbus register to be read must be specified. Types of data available:

  - **Uint16:** Unsigned 16-bit integer (1 WORD)
  - **Uint32:** Unsigned 32-bit integer (2 WORDS)
  - **Int16:** 16-bit integer (1 WORD)
  - **Int32:** 32-bit integer (2 WORDS)
  - **Int64:** 64-bit integer (4 WORDS)
  - **Float:** Float (32 bit, 2 WORDS)
  - **Double:** Double (64 bit, 4 WORDS)
  - **String:** String (X WORDS)
  - **Bit:** Bit (1 WORD)

- ***Flip:*** Enables the rotation of words or bytes of a particular data type, allowing all combinations.

  - **No:** No rotation takes place.
  W Rotation of words from the register.
  - **B:** Byte rotation of each word.
  - **W+B:** Rotation of words and bytes.

  Example of rotation for a **DOUBLE**-type register:

No:	AB CD EF GH
W:	GH EF CD AB
B:	BA DC FE HG
W+B:	HG FE DC BA

- ***Com.:*** Specifies the register read command

- **1:** Coil-type register readings
- **2:** Bit-type register readings (discrete inputs)
- **3:** Holding register readings
- **4:** Input register readings

- ***name:*** (Optional). Text with the register name.

- ***name:*** (Optional). Text with the register units.

- **mode:** Tells you what to do with the register. The following options are available:

- Not used: The register will not be read by the router.
- Not saved: The register will be read, but the reading will not be stored in the datalogger. This can be useful for certain types of registers that are dependent on others. For example, if an X-register only indicates whether a third register uses volts or millivolts as units, you may need to read it, but you won't need to store the information in the datalogger. In this case, "Not saved" mode is the most appropriate.
- Instant: The register will be read, and when the time indicated in the period is reached, the value read at that time (instantaneous value) will be the one stored in the datalogger.
- Average: The record will be read, and when the time indicated in the period is reached, the average value recorded during the reading period is the one that will be stored in the datalogger.
- Minimum: The register will be read, and when the time indicated in the period is reached, the minimum value read during the reading period is the one that will be stored in the datalogger.
- Maximum: The register will be read, and when the time indicated in the period is reached, the maximum value read during the reading period is the one that will be stored in the datalogger.

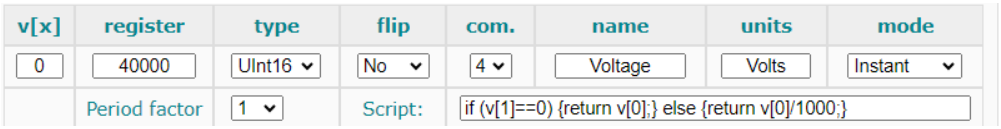
NB: the number of samples taken to generate the average values will vary depending on the number of devices and registers to be read. The Titan router will read all registers cyclically at the highest possible speed. The higher the number of devices and the higher the number of readings to be taken, the lower the number of samples.

- ***Period factor:*** A numeric field for setting the period multiple for readings. For example, if the "Period" field is assigned a value of 5, register readings will be stored in the datalogger every 5 minutes by default. But if a certain register is to be stored every 15 minutes, a "3" should be entered in the "Period factor" field. In this case, data would be stored in the datalogger 1 out of 3 times every 5 minutes, i.e. every 15 minutes.

- ***Script:*** (Optional). A short javascript script (v5) that can be entered to operate on a value read. This enables access to both the v[x] values and the "mtx" object (see TITAN SCRIPTS section for more information on the "mtx" object).

Examples:

In the case of the script "**return v[0]/1000;**", the value stored in the datalogger would not be the value read directly from the register 40000; however, before storing it in the datalogger, it would be divided by 1000 to store the value in volts.

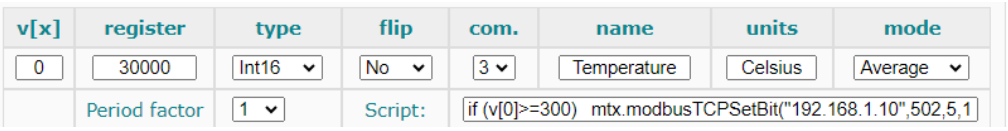


In the case of the script:

```
If (V[1]==0) {
        return v[0];
else
        return v[0]/1000;
}
```

the value stored in the datalogger would not be the value read directly from the register 40000; however, before storing it, if register v[1] had the value "0", the value of v[0] would be stored directly. Meanwhile, if v[1] had the value "1", the value v[0]/1000 would not be stored;

In other words, the scripts allow operations to be performed on the registers before they are stored in the datalogger, even enabling interaction with the value of other registers.



Let's imagine that with this configuration, the average temperature is stored in register 30000, v[0], which the sensor returns in tenths of a degree. For example, it will return 300 for 30.0ºC, 252 for 25.2ºC, etc. The aim is for the average temperature to be stored in the datalogger, but in degrees Celsius, so you will need to include a "**return v[0]/10**"; in the script.

If the temperature is equal to or greater than 30ºC, you will need to write a value of "1" in the Modbus 10000 bit-TYPE register of a PLC operating with modbus TCP in order to activate a fan. However, if the temperature is below 29°C, you will need to write a "0" in the 10000 register to stop the fan.

In this case, the final script would look like this:

```
If (V[0]==300)
        mtx.modbusTCPSetBit("192.168.1.10",502,5,10000,1);
elseif (v[0]<=290)
mtx.modbusTCPSetBit(&amp;quot;192.168.1.10&amp;quot;,502,5,10000,0);
return v[0]/10;
```

NB: It is possible to reference the value of another register from a register script. For example, you can reference the v[0] register from the v[1] register script. The v[0] references the register value without running through the script that may be present in the v[0] register. In other words, if there is a script that says “return v[0]*2;” in register v[0], references to v[0] from the v[1] script will be made to the value of v[0] **before** it is multiplied by 2. If you wish to obtain the value of v[0] **after** it has run through the script (i.e. the value multiplied by 2), you must refer to **vs**[0] instead of **v**[0] in the v[1] script.

Datalogger storage format.

When Modbus registers are stored in the datalogger memory, they are stored in the following JSON format, as shown in the following example:

{"IMEI":"865583042283167","TYPE":"MODB2","TS":"2023-10-03T11:46:00Z","P":"TITAN","ID":"Device2","A":"2","data":[{"R":30000,"V":"30.0","N":"

Temperature","M":"3","U":"Celsius","S":"OK"},{"R":30001,"V":"40.0","N":"Humidity","M":"3","U":"%","S":"OK"}]}

Where:

IMEI: router IMEI

TYPE: Frame type

TS: Timestamp of when the Modbus registers were read

P: Field “ID” of the “Logger” configuration section

ID: “ID” field of the Modbus device section

A: Modbus device address

R: Register address

V: Register value

N**:** Register name

M: Register mode (2: instantaneous value, 3: average value, 4: max value, 5: min value)

U: Units

S: Reading status (OK: correct reading, ERR: incorrect reading)

Additional Notes.

- We recommend you read the application note **"ANV6_72-Router-Titan-Modbus-Expert"** to further your understanding.

4.6.4- External Devices → GPS Receiver

This menu allows you to configure the GPS positioning data of TITAN-based devices using an internal GPS module or, if no internal GPS module is available, a NMEA GPS receiver connected to one of its RS232 or RS485 ports

- ***Enabled***: Check this box if you have an internal (or external) GPS module connected to the TITAN-based device you wish to use.
- ***Serial Port***: allows you to select the serial port of the TITAN-based device where the GPS device is connected.
- ***Interval***: select the GPS reading period as a value of x seconds (you will receive a GPS position reading every x seconds). Indicate a minimum value of 30 seconds.
- ***Logger***: check this box if you wish to record the GPS positions read in the internal logger, for example, to send them to your web platform via HTTP/HTTPS, MQTT/MQTTS or FTP. Very useful if you wish to use TITAN-based devices to implement a GPS fleet management system.
- ***System time***: check this box if you would like the system time to be synchronised with the time read from the GPS.

- ***TCP port listener***: allows NMEA frames to be obtained through an IP connection made via the TITAN-based device. For example, if you open and connect a socket to the IP of the TITAN-based device to the port specified in this point, you will be able to obtain real-time NMEA positioning frames.

- ***TCP Client IP***: allows you to specify an IP address in order to send NMEA frames to an external server.
- ***TCP Client Port***: allows you to specify the TCP port to send NMEA frames to from the external server specified in the "TCP Client IP" parameter.
- ***TCP Client Timeout***: specifies the number of seconds the TITAN-based device should allow to elapse between detecting the closure of the socket and reopening it again.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.6.5- External Devices → Generic Serial Device

In this section, you will be able to configure an RS232 or RS485 serial device datalogger. Essentially, the TITAN-based device will read, store the serial data collected by any of the above mentioned interfaces and send it to a web platform via HTTP/S, MQTT/S or FTP.

This is particularly suitable for collecting data from serial devices that produce a data frame every so often (temperature sensors, humidity sensors, serial alarms, etc.)

- ***Enabled***: check this box if you wish to connect a serial device and have the TITAN-based device perform the datalogger function.
- ***Serial Port***: allows you to select the serial port of the TITAN-based device where the serial device to be logged is connected.
- ***Interval***: Specifying "0" will collect all received serial frames. Specifying "1" will pick up every second frame received. Specifying "2" will pick up every third frame received.
- ***Only changes***: check the box if you only wish to log frames that differ from the previous frames. For example, if you connect a temperature sensor, that sensor may be producing the same serial data for quite some time until the temperature changes. If you check this box, the serial data will only be collected when the data changes.
- ***Logger***: check this box if you wish to save the serial data in the internal logger in order to send it later to your web platform via HTTP/HTTPS, FTP or MQTT/MQTTS
- ***TX period***: a number greater than 0 indicates the period (in seconds) within which the "TX Frame1", "TX Frame2", "TX Frame3", "TX Frame4", "TX Frame5" data frames are sent via the serial port. This option allows you to "interrogate" serial devices with proprietary protocols.
- **TX FrameX**: indicates a data frame to be re-sent via the serial port every X seconds specified in the "TX period" parameter. The serial frames must be entered in hexadecimal format. Example: 0102030A0D…

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.6.6- External Devices → IEC102 Meter

In this section, you can configure the TITAN-based device to read an electricity meter with the IEC 60870-5-102 protocol.

- ***Enabled***: Check this box if you wish to connect an electricity meter with the IEC- -60870--5-102 protocol, to which the TITAN-based device must interrogate autonomously.
- ***Communication Port***: Select the serial port of the TITAN-based device where the IEC-60870-5-102 electricity meter is connected.
- ***Period***: period (in minutes) in which the IEC-60870-5-102 electricity meter shall be interrogated by the TITAN-based device to read the instantaneous values.

- ***ID Meter***: enter an identifying text for the electricity meter.
- **Link Address**: link address of the electricity meter.
- **Meas. Address**: address of the measurement point of the electricity meter.
- **Password**: electricity meter password.

- **Sign**: check the box if you want the parameters PAT/FPT, PAF1/FPF1, PAF2/FPF2 and PAF3/FPF3 to contain a negative value if energy is exported.
- **Optical probe**: check the box if you are using an echoed optical probe (if you do not check this box and the optical probe is echoed, the meter reading will not work).

- ***Fiscal close***: check this box if, in addition to reading the instantaneous values, the TITAN-based device must also read the closing values. If the box is checked, the TITAN-based device will read and store the meter closing values at midnight every evening.
- ***Fiscal close days***: indicates the number of days for the reading of closing values. By default, 31 days.

- ***Logger***: check this box if you wish to save the serial data in the internal logger in order to send it later to your web platform via HTTP/HTTPS, FTP or MQTT/MQTTS.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.
- If you also need to read the integrated totals (load curves), you will have to do so from the scripts section.
- The frames sent by the TITAN-based device will take the following format:

Example of a frame stored in the Logger with the instantaneous values:

{"**IMEI**":"867962046823806","**TYPE**":"IEC102","**TS**":"2021-11-02T15:14:04Z","**P**":"1234","**ID**":"ID00000","**VABA**":0,"**VABRI**":0,"**VABRC**":0,"**PAT**":0,"**PRT**":0,"**FPT**":1000,"**PAF1**":0,"**PRF1**":0,"**FPF1**":1000,"**PAF2**":0,"**PRF2**":0,"**FPF2**":1000,"**PAF3**":0,"**PRF3**":0,"**FPF3**":1000,"**IF1**":0,"**TF1**":1337,"**IF2**":0,"**TF2**":1108,"**IF3**":0,"**TF3**":3}

Where:

IMEI: ID number of the TITAN-based device.

TYPE: JSON type. In this case the type is IEC102.

TS: timestamp of the time the data was collected

P: optional identification field (of the Logger)

ID: meter identifier

VABA: absolute active energy

VABRI: inductive absolute reactive energy

VABRC: inductive absolute reactive energy

PAT: total active power

PRT: total reactive power

FPT: total power factor

PAF1: phase I active power

PAF1Dir Imported (0) / Exported (1)

PRF1: phase I reactive power

FPF1: phase I power factor

PAF2: phase II active power

PAF2Dir Imported (0) / Exported (1)

PRF2: phase II reactive power

FPF2: phase II power factor

PAF3: phase III active power

PAF3Dir Imported (0) / Exported (1)

PRF3: phase III reactive power

FPF3: phase III power factor

IF1: phase I intensity

TF1: phase I voltage

IF2: phase II intensity

TF2: phase II voltage

IF3: phase III intensity

TF3: phase III voltage

- See Application Note ANV6_51-Router-TITAN-Metering-IEC-60870-5-102.pdf

Example of a frame stored in the Logger with the closing values:

{"**IMEI**":"869101057093607","**TYPE**":"IEC102_CTAVM2","**TS**":"09/06/2023 08:46:23",”**P**”:”1234”,"**ID**":"ID00000","**CTAVM2**":

[

{"**DO**":20,"**EaA**":0,"**EiA**":0,"**CA**":2,"**EaRi**":0,"**EiRi**":0,"**CRi**":2,"**EaRc**":0,"**EiRc**":0,"**CRc**":2,"**R7**":0,"**C7**":128,"**R8**":0,"**C8**":128,"**MPA**":0,"**FMPA**":"2023-01-09T12:58-0","**CMA**":2,"**EPA**":0,"**CE**":128,"**DINI**":"2022-12-01T00:00-0","**DEND**":"2023-01-09T12:58-0"},

{"DO":21,"EaA":0,"EiA":0,"CA":2,"EaRi":0,"EiRi":0,"CRi":2,"EaRc":0,"EiRc":0,"CRc":2,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":2,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},

{"DO":22,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},

{“DO”:23,”EaA”:0,”EiA”:0,”CA”:0,”EaRi”:0,”EiRi”:0,”Cri”:0,”EaRc”:0,”EiRc”:0,”CRc”:0,”R7”:0,”C7”:128,”R8”:0,”C8”:128,”MPA”:0,”FMPA”:”2023-01-09T12:58-0”,”CMA”:0,”EPA”:0,”CE”:0,”DINI”:”2022-12-01T00:00-0”,”DEND”:”2023-01-09T12:58-0”},

{"DO":24,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},

{"DO":25,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},

{"DO":26,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"}

]

}

Where:

IMEI: ID number of the TITAN-based device.

TYPE:	JSON type. In this case the type is IEC102_CTAVM2.
TS:	timestamp of the time the data was collected
P:	optional identification field (of the Logger)
ID:	meter identifier
CTAVM2:	data array with the readings obtained
DO:	object address
EaA:	absolute active energy
EiA:	absolute active energy
CA:	active energy qualifier
EaRi:	inductive absolute reactive energy
EiRi:	inductive incremental reactive energy
Cri:	inductive reactive energy qualifier
EaRc:	capacitive absolute reactive energy
EiRc:	capacitive incremental reactive energy
CRc:	capacitive reactive energy qualifier
R7:	register 7 reserve
C7:	register 7 reserve qualifier
R8:	register 8 reserve
C8:	register 8 reserve qualifier
MPA:	maximum power
FMPA:	date of maximum
CMA:	maximum qualifier
EPA:	power excesses
EC:	excess qualifier
DINI:	start of period
DEND:	end of period

4.7.1- Other → AT Command

In this section, you can send an AT command directly to the internal modem of the TITAN-based device. For example, you may wish to check the coverage or to identify nearby telephone cells, etc.

It is also possible to configure up to 5 special AT commands through which to configure the device at start-up.

- ***AT Command:*** AT command for real-time execution (e.g.: AT+CSQ). Once you click on the "SEND AT COMMAND" button, the AT command will be executed.
- ***AT1, … AT5:*** AT initialization commands.

Additional Notes.

- Once the AT initialisation commands have been created, click on the "SAVE CONFIG" button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.7.2- Other → DynDns

TITAN-based devices are compatible with the DynDNS and No-IP service. If you do not have a SIM card with a fixed IP and you are interested in using the external DynDNS or NO-IP services, you can configure them in this section.

- ***Enabled:*** Check this box if you wish to enable the use of DynDNS or NO-IP.
- ***Server:*** Specify the service server (members.dyndns.org or dynupdate.no-ip.com).
- ***Domain:*** enter the dns you have created (e.g.: mydomain.dyndns.org)
- ***Login***: login for your DynDNS or NO-IP account
- ***Password***: password for your DynDNS or NO-IP account
- ***Period***: period, in minutes, in which the current IP is refreshed on the DynDNS or NO-IP servers
- ***Hide WAN IP***: if the box is checked, the IP will be detected by the DynDNS server. You must activate this if you wish to use DynDNS, using the Ethernet or WiFi port of the TITAN-based device as an Internet connection.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

- The IP is updated on the DynDNS or NO-IP servers every time it changes. However, we recommend you use the configuration parameter "Period", e.g.: at a value of 60, so that whatever happens, it is sent every hour.

4.7.3- Other → Private DynDns

The Private DynDNS service enables the current IP address of the TITAN-based device and certain status values to be sent to its own server. Data can be sent either via HTTP (HTTP/S GET, HTTP/S POST) or by sending a specific frame via a socket to a configurable TCP port and also via MQTT.

HTTP or TCP Socket Method

- ***Enabled:*** check this box if you wish to enable the use of Private DNS
- ***Mode:*** you can choose between TCP socket and HTTP/S GET, HTTP/S POST and HTTP/S PUT
- ***Server:*** IP or DNS of the remote server
- ***Server Login***: login of your web server (if you use a "HTTP" mode)
- ***Password***: application password of your web server (if you use a "HTTP" mode)
- ***TCP Port***: TCP port if using "socket" mode
- ***ID***: identifier string (for either "http get" or "socket" mode)
- ***Period***: The period, in minutes, in which the current IP is refreshed on your server. Please bear in mind that regardless of this period, as soon as the public IP changes, it is sent.
- ***Custom header1:*** Custom HTTP header 1
- ***Custom header2:*** Custom HTTP header 2
- ***Custom header3:*** Custom HTTP header 3
- ***IO Status***: this must be activated if you also wish to send the status of digital inputs and outputs and meters.

MQTT method

- ***Enabled:*** check this box if you wish to enable MQTT
- ***Period***: period, in minutes, in which the current IP will be sent to your server via MQTT. Please bear in mind that regardless of this period, as soon as the public IP changes, it is sent.
- ***MQTT Topic:*** MQTT Topic to be used for reporting with the IP
- ***MQTT QoS***: quality of service used by MQTT to send the IP
- ***IO Status***: this must be activated if you also wish to send the status of digital inputs and outputs and meters.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.
- Example of the JSON format frame sent:

{"TYPE":"DNS","IMEI":"867962046823806","IP":"88.28.221.24","P":"","CSQ":24,"MOD":"","VER":"5.2.6.02","IMSI":"214075536243578","TECH":"4g","TS":"2021-11-02T16:00:51Z","CID":"214;07;21E0;13B6D0A","RSSI":"-65","RSRP":"-91","RSRQ":"-6","IO0":0,"IO1":0,"IO2":0,"IO3":0,"IO4":0,"CO0":0,"CO1":5,"CO2":0}

Where:

TYPE: type of frame. In this case, DNS.

IMEI: device identification number. Unique for each device

P: Logger ID field (External Devices > Logger configuration)

IP: IP of WAN interface (2G / 3G)

CSQ: 0...31 (signal strength)

MOD: TITAN-based device model

See: FW Version

IMSI: IMSI of the SIM card

TECH: technology used (2G/3G/4G)

TS: timestamp

CID: identification of telephone cell used

RSSI: signal strength level.

RSRP: 3G rsrp

RSRQ: 3G rsrq

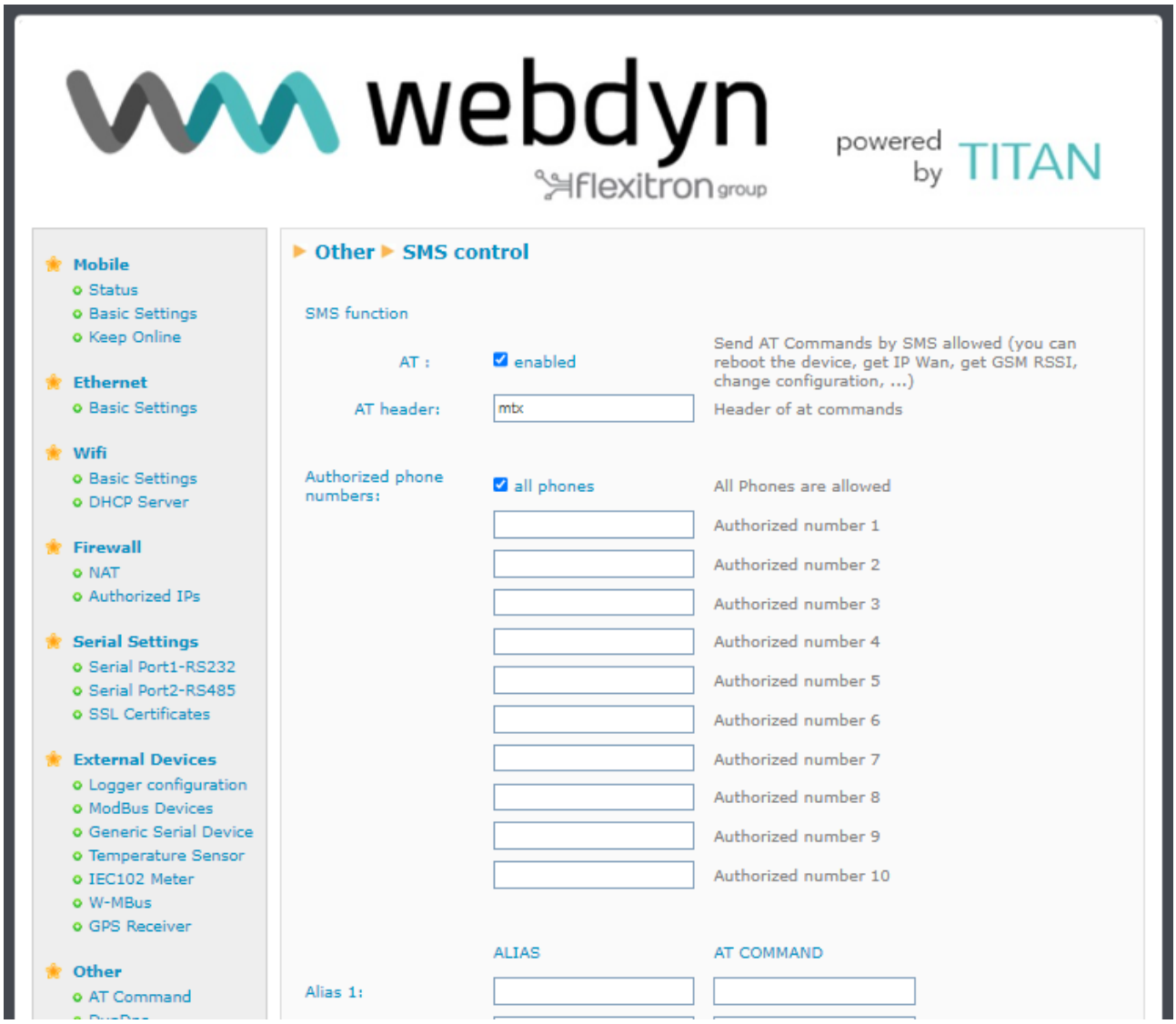
IOx: current value of digital input / output "x"

COx: current value of the pulse meter associated with digital input "x"

4.7.4- Other → SMS Control

This section lets you configure control of the device using SMS messages. For example, you can configure this section so that the device restarts or changes a digital output when an SMS is received, or to specify the telephone numbers authorised for this purpose.

- ***AT enabled:*** check this box if you wish to send AT commands to the TITAN-based device via SMS, e.g.: to find out the coverage remotely, to perform a re-set or to change the configuration, etc.
- ***AT header****:* here you can enter the header text for SMS command messages. For example, if you type the text “mtx” in this box, when an AT command is sent by SMS, e.g.: the “AT+CSQ” command to find out the general coverage level, you must send an SMS message with the following text “mtx AT+CSQ”
- ***All phones****:* check this box if you want all telephones to be able to send AT commands to the device. Do not check this box if you want to specify a set of authorized phone numbers.
- ***Authorized Number X:*** in these boxes you can specify up to 10 authorised telephone numbers.
- ***Alias / ATCommand***: up to 10 aliases can be entered to execute SMS commands. Imagine you wish to send an SMS to update a certain MODBUS register of an external device. For example, you can configure an ALIAS so that when the TITAN-based device receives the text "reg on", it will internally execute the AT command: at^mtxtunnel=setmodbus,1;5;16;2 by writing the value 2 in register 5 of the device with Modbus address 1.
- ***Alias Result OK:*** text that is sent in response when the execution of an ALIAS command is successful. If you wish, you can return a customised response for each ALIAS, indicating the response between tags: <a1>Ok</a1><a2>Perfect</a2>...
- ***Alias Result ERROR:*** text that is sent in response when the execution of an ALIAS command is wrong. If you wish, you can return a personalised response for each ALIAS, indicating the response between tags: <a1>Error</a1><a2>Upssss</a2>...



Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.
- See application note ANV6_3 for further information through examples. You will also see how to add parameters to the ALIAS

4.7.6- Other → Periodic auto-reset

In this section you can configure a scheduled auto-reset for the TITAN-based device.

- ***Autoreset not enabled***: enable this option if you do not want the TITAN-based device to reset itself periodically.
- ***Autoreset every X hours***: enable the option if you want the TITAN-based device to auto-reset every X hours.
- **Number of hours**: if you select autoreset every X hours, you must specify X, i.e. the number of hours after which the reset takes place, in this box. Specify 24 for a daily reset.
- ***Auto-reset at specific time***: select this option if you want the TITAN-based device to auto-reset at a particular time of day.
- ***Time for autoreset***: specific time for daily auto-reset.
- ***Auto-reset if router can't obtain IP after X minutes***: highly recommended for context loss situations. For example, it lets you specify the number of minutes after which the TITAN-based device will reset itself if it cannot obtain an IP address.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.7.7- Other → Time Servers (NTP)

The TITAN-based device has a built-in, real-time clock that enables it to keep the time even if power is lost for several hours. This built-in clock must be synchronised periodically with time servers via the NTP protocol.

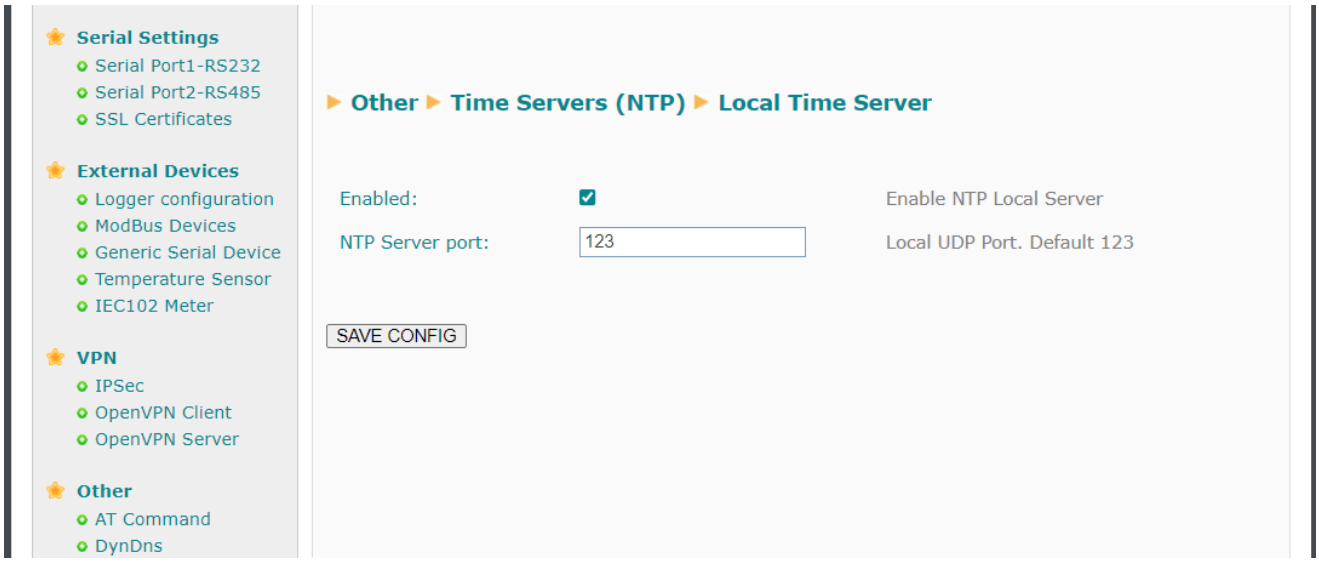
Time Servers (NTP)

- ***Enabled:*** check this box if you want to use NTP time servers.
- ***NTP Server 1:*** IP or DNS address of the NTP 1 time server
- ***NTP Server 1 port:*** port of the 1 time NTP server
- ***NTP Server 2***: IP or DNS address of the NTP 2 time server
- ***NTP Server 2 port:*** port of the 2 time NTP server
- ***Time Zone****: lets you specify the time zone.*

Time Servers (NTP) → Local Time Server.

- ***Enabled:*** check this box if you would like to use the Titan-based router as an NTP time server.
- ***NTP Server port***: NTP server port of the Titan-based router

It is also possible to configure the Titan-based router as an NTP time server. To do this, simply check the “Enable” box and select the desired UDP port (by default, 123). It is useful, for example, to set the time correctly on devices even without using a SIM card (as long as it is a GPS-enabled Titan model).



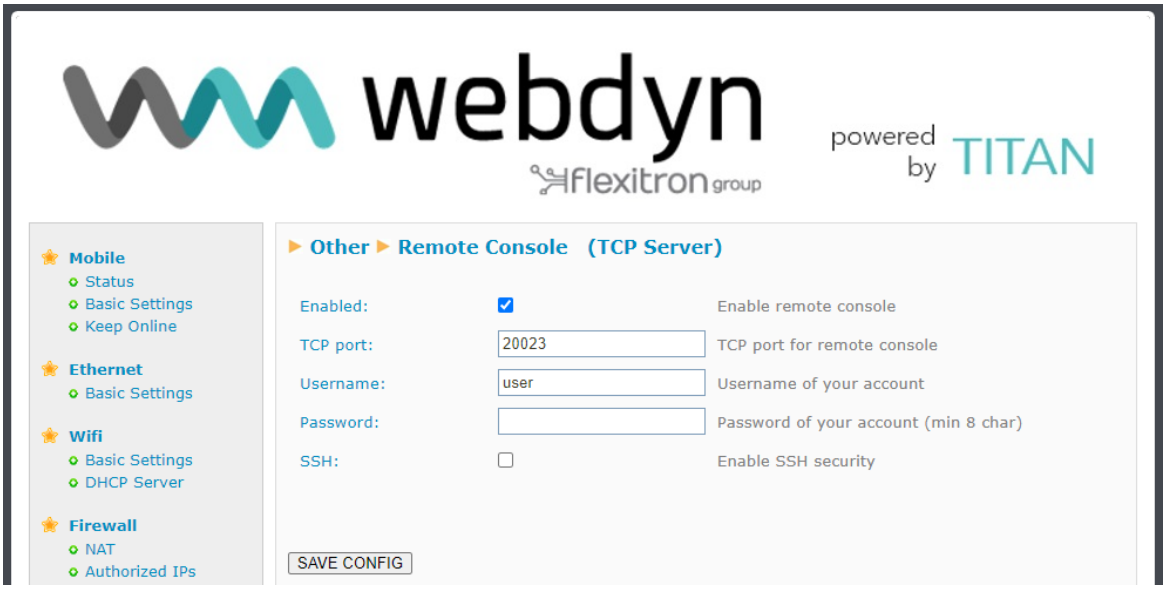
Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.7.8- Other → Remote console (TCP Server)

If at any time you need to perform a special operation using the TITAN-based device, the “Telnet or “SSH” connection can be configured in this section. In other words, this special connection also makes it possible to send AT commands to the TITAN-based device via a TELNET or SSH connection. This lets you change the configuration of the device, switch a digital output, etc. See section 5 of this manual for a list of available AT commands.

- ***Enabled:*** check this box if you want to use this special connection.
- ***TCP Port***: listening TCP port of the device where the connection must be made.
- ***Login:*** username (will be requested after establishing the connection).
- ***Password:*** user password (will be requested after entering the username)
- ***SSH:*** check the box if you wish to use SSH instead of Telnet.



Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect. **If you check/uncheck the SSH box, you will also need to re-enter the password.**
- The remote console can be accessed both locally (Ethernet or Wifi) and remotely via a 4G/3G/2G connection.

4.7.9- Other → SNMP

TITAN-based devices have an SNMP protocol. You can perform SET and GET operations from standard SNMP applications through SNMP

- ***Enabled:*** Select whether you wish to enable the SNMP service of the TITAN-based device
- ***SNMP Version:*** You can choose between SNMPv2c and SNMP v3
- ***UDP port:*** The standard UDP port for SNMP is 161, but you can specify a custom one.
- ***Custom OID:*** lets you change the default value for the Enterprise-Product OID (.45711.1.1) if you wish to adjust it to match your corporate values.
- ***Community:*** password to execute SET and GET commands. Only required for SNMPv2
- ***Username:*** username when using SNMPv3 (not required for SNMPv2)
- ***Auth Password:*** authentication password for SNMPv3 (not required for SNMPv2)
- ***Priv Password:*** privacy password for SNMPv3 (not required for SNMPv2)
- ***Auth Protocol:*** authentication protocol (MD5 or SHA)
- ***Priv Protocol:*** encryption protocol (DES, AES128)
- ***Traps - Enabled:*** enables the traps of the TITAN-based device
- ***Traps - UDP Port:*** lets you specify the port for SNMP TRAPS.
- ***Traps - IP:*** IP address for sending SNMP TRAPS.
- **Traps - Community:** community field for sending TRAPS (only for SNMPv2c)
- **Alarm OS:** enables the alarm traps of the device's operating system in the event of a critical failure.
- **Number traps alarm on:** Indicates the number of traps to send when an alarm trap is activated.
- **Number traps alarm off:** Indicates the number of traps to send when an alarm trap is activated.
- **Trap period:** Indicates the period, in seconds, between sending alarm traps.

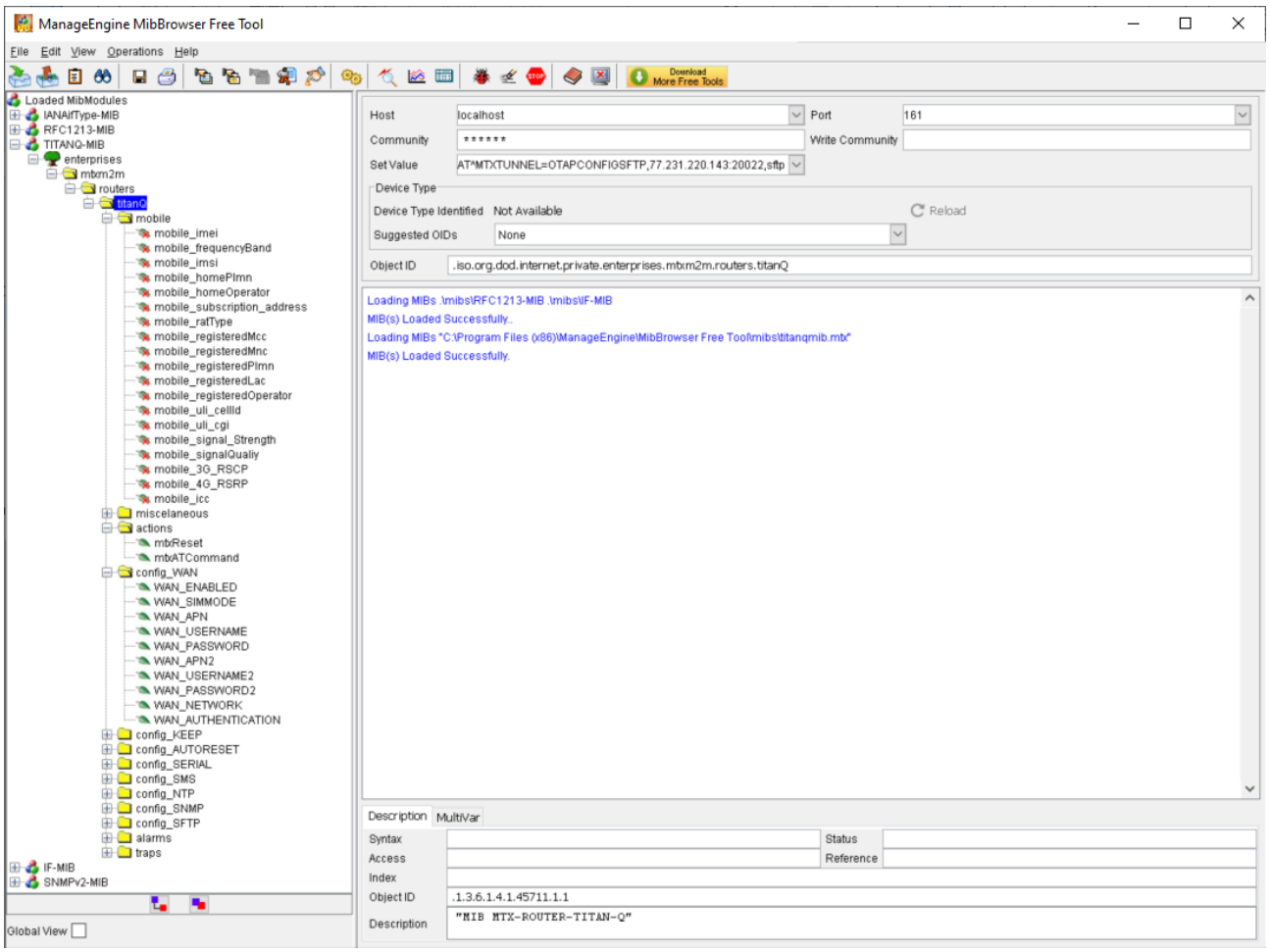
Additional Notes.

- Once the configuration process is complete, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

- The previous screen has a link labelled “Click here to download MIB”. Download the file to obtain the MIB with the OIDs.

- You can use SNMP to monitor the device, for example, by interrogating the device to obtain the RSSI (coverage), the date and time, the uptime, whether it is operating in 4G/3G/2G mode, or the operator used.

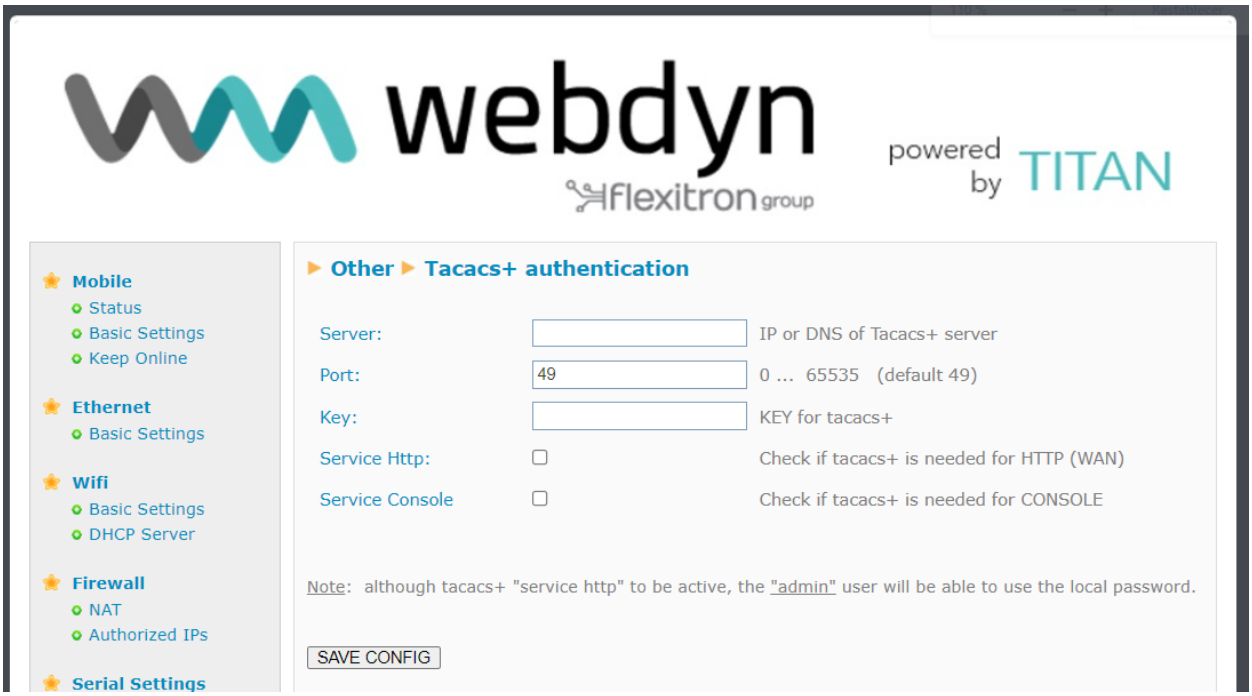
- The "Action" section offers very interesting options. You can do 2 things in this section. One of them is to reset the device remotely. The other is to execute an AT command remotely via SNMP (writing in the register will execute the AT command and reading it will give you the response). By sending AT commands you can do anything via SNMP, from reading a Modbus device connected to the device, changing configuration parameters, reading a GPS position, reading a digital input, changing the status of a digital output, etc.



4.7.10- Other >> TACACS+

TITAN-based devices can be authenticated using Tacacs+. This feature enables external authentication for HTTP, Telnet and SSH services of TITAN-based devices. You must configure this section to use this feature.

- ***Server:*** IP or DNS address of the Tacacs+ server
- ***Port:*** listening port of the Tacacs+ server (by default, 49).
- ***KEY:*** encryption password.
- ***Service http:*** check the box if you wish to use the Tacacs+ authentication service for HTTP access to the TITAN-based device.
- ***Service Console:*** check the box if you wish to use the Tacacs+ authentication service for TELNET SSH (Remote Console) access to the TITAN-based device.



Additional Notes.

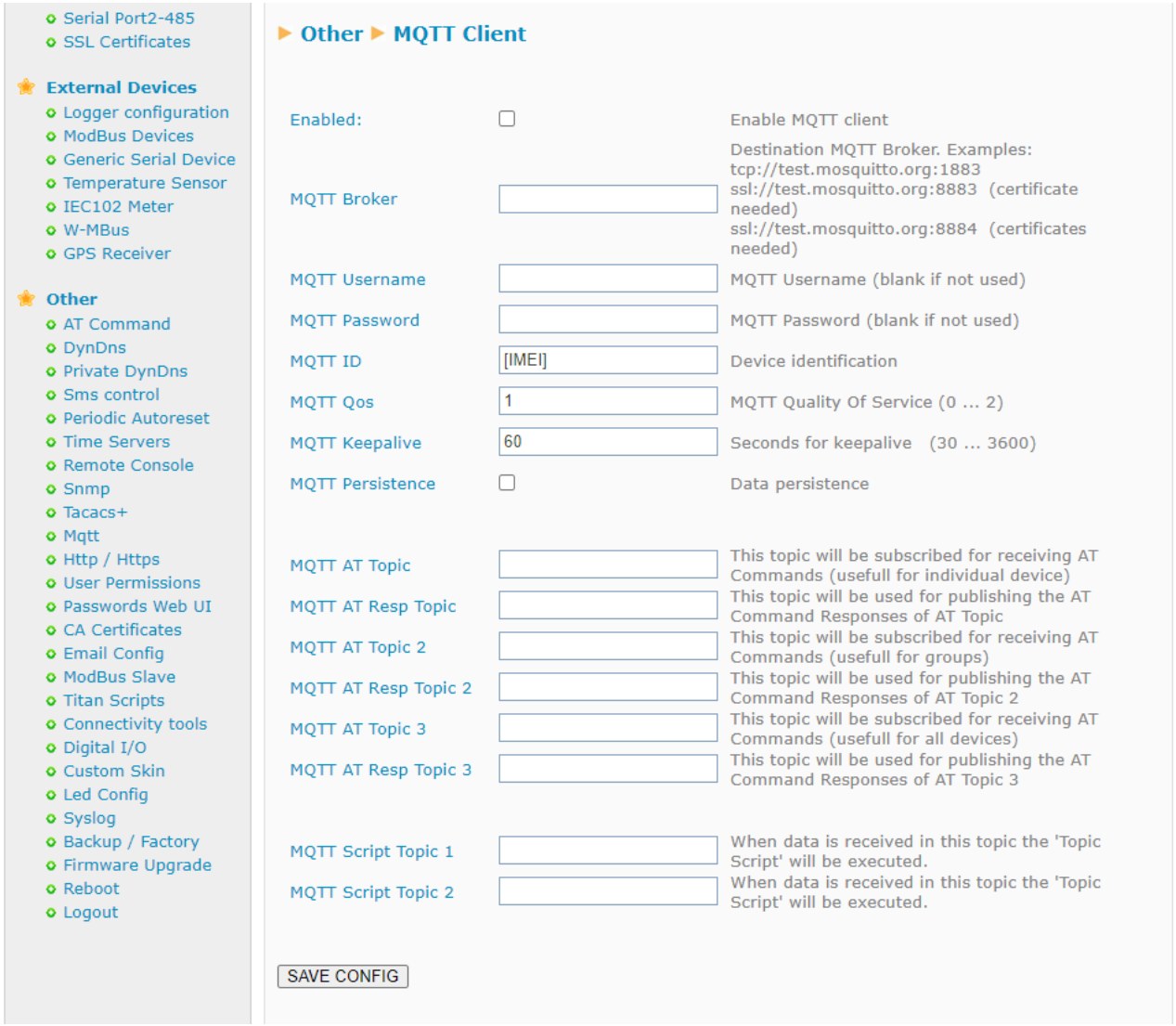
- Once the configuration process is complete, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.7.11- Other → MQTT

TITAN-based devices can operate as an MQTT client by connecting to an MQTT / MQTT broker. Configuring the TITAN-based device as an MQTT client lets you send the data collected in the internal datalogger (data from sensors, Modbus devices, etc.) via MQTT. You must properly configure this section if you select sending by MQTT in the LOGGER section.

- ***Enabled:*** Select whether you wish to enable the MQTT Client service
- ***Username:*** Username MQTT. Blank if not used.
- ***Password:*** Password MQTT. Blank if not used
- ***ID:*** Device identification field
- ***QoS:*** Quality of service (0,1,2)
- ***KeepAlive:*** Seconds for keepalive (recommended 300)

- ***Persistence:*** Select whether you want the data to be continuous
- ***AT Topic:*** Topic 1 to which TITAN-based devices will subscribe. AT commands sent to this topic will be executed on the device.
- ***AT Resp Topic***: Responses to AT commands received via MQTT by TITAN-based devices in the "AT Topic" will be sent to this Topic via MQTT.
- ***AT Topic 2:*** Topic 2 to which TITAN-based devices will subscribe. AT commands sent to this topic will be executed on the device.
- ***AT Resp Topic 2:*** Responses to AT commands received via MQTT by the TITAN-based devices in the "AT Topic 2" topic will be sent to this Topic via MQTT
- ***AT Topic 3:*** Topic 3 to which TITAN-based devices will subscribe. AT commands sent to this topic will be executed on the device.
- ***AT Resp Topic 3:*** Responses to AT commands received via MQTT by the TITAN-based devices in the "AT Topic 3" topic will be sent to this Topic via MQTT
- ***MQTT Script Topic 1:*** Topic MQTT in which a customisable script will be executed when receiving data. See the scripts section of this manual and the application notes for more information.
- ***MQTT Script Topic 2:*** Topic MQTT in which a customisable script will be executed when receiving data. See the scripts section of this manual and the application notes for more information.
- ***Client Certificate:*** When using MQTTS with client authentication you will need to enter the .key file of the client certificate (PEM format) in this field.



Additional Notes.

- Once the configuration process is complete, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

- Please note that it is possible to enter the text [IMEI] instead of the IMEI number. That is, if the IMEI is 012345678912345, that would be the same as using TOPIC /0123456789012345/TEST as the topic /[IMEI]/ TEST

- Please note that if you use MQTTS, you may need to include the CA certificate used by the MQTT broker in the "Other → CA Certificates" section.

- In previous TITAN firmware versions the response to AT commands in any topic was sent to a single topic. If you wish to see the same behaviour, you can set the same value in the topics "AT Resp Topic", "AT Resp Topic 2” and "AT Resp Topic 3"

4.7.12- Other → HTTP / HTTPS

HTTP configuration and HTTPS activation for the configuration environment.

- ***HTTP Port:*** indicates the TCP port for HTTP remote configuration. For example, if you specify 8080, the configuration URL will be http://x.x.x.x:8080 . By default, the standard port is 80, but if you wish to perform an NAT to the TCP80 port of an internal ETH device (IP camera, PLC), you will need to change it, for example, to 8080. The HTTP service is not remotely usable if HTTPS is enabled.
- ***HTTPS Enabled***. enables the HTTPS service (self-signed certificates are generated automatically after restarting).
- ***HTTP Port:*** indicates the TCP port for HTTPS configuration. By default, 443.
- ***HTTPS Key and Cert:*** You can use your own certificate for the TITAN-based device's HTTPS server. In this field you can enter, in PEM format, the CERTIFICATE and the KEY in the same file (PEM format)

Additional Notes.

- Once the configuration process is complete, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

- When trying to connect to the TITAN-based device via HTTPS, the browser will probably show a warning message about the self-signed digital certificate. This is normal.

4.7.13- Other → User Permissions.

In this section the user can configure the permissions to which the “user” and “guest” users will have access. Configuration options that are not selected will not appear in the left-hand menu of the configuration environment when logging in to the device using the username “user” or “guest”.

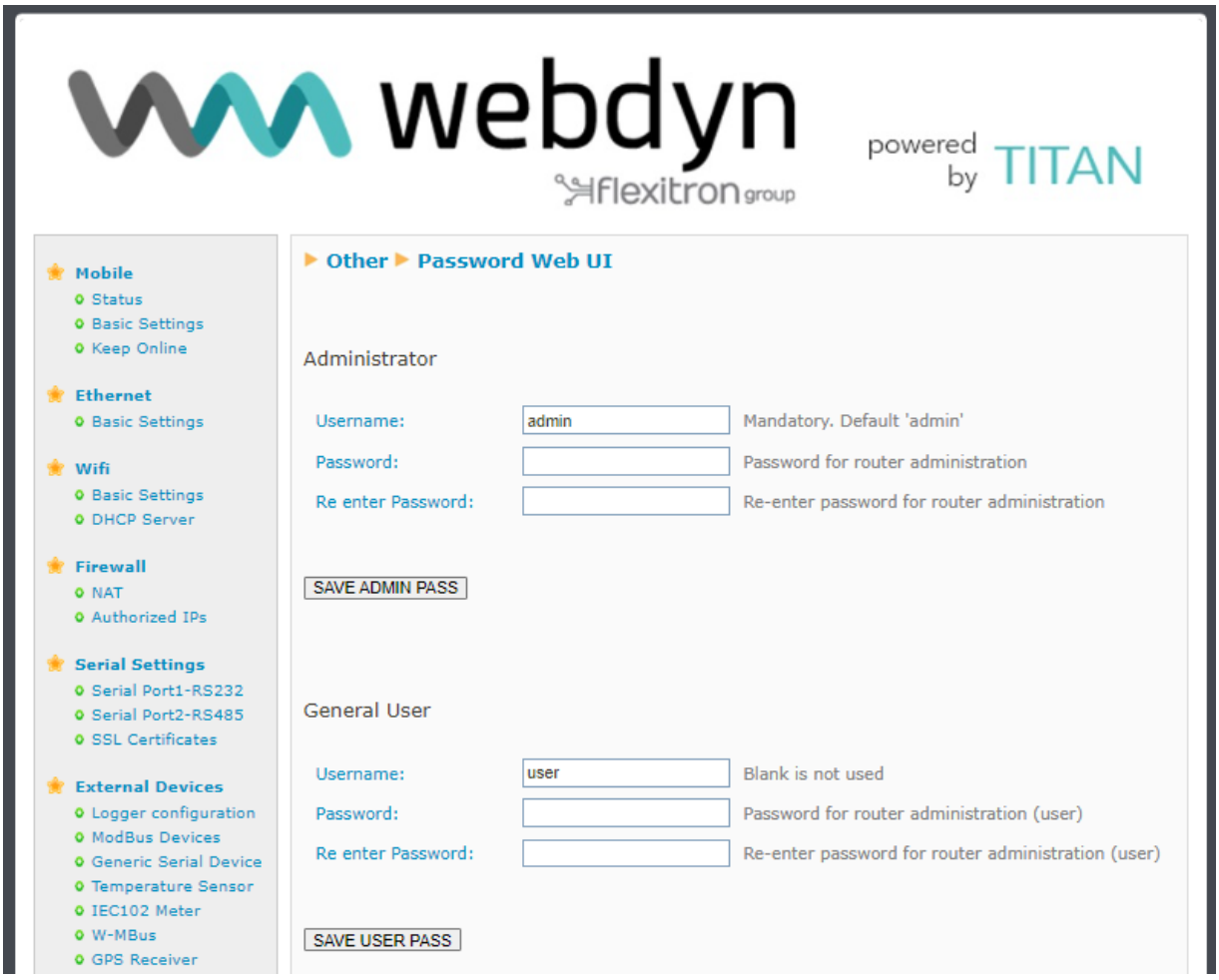

webdyn
Flexitron group
powered by TITAN
Mobile
Status
Basic Settings
Keep Online
Ethernet
Basic Settings
Wifi
Basic Settings
DHCP Server
Firewall
NAT
Authorized IPs
Serial Settings
Serial Port1-RS232
Serial Port2-RS485
SSL Certificates
External Devices
Logger configuration
ModBus Devices
Generic Serial Device
Temperature Sensor
IEC102 Meter
W-MBus
GPS Receiver
Other
AT Command
DynDns
Private DynDns
Sms control
Periodic Autoreset
Time Servers
Remote Console
Snmp
Tacacs+
Mqtt
Http / Https
User Permissions
Passwords Web UI
CA Certificates
Email Config
ModBus Slave
Titan Scripts
Connectivity tools
Digital I/O
Custom Skin
Led Config
Syslog
Backup / Factory
Firmware Upgrade
Reboot
Logout
Other ▸ User Permissions
Mobile
Basic Settings
Keep Online
Ethernet
Basic Settings
Wifi
Basic Settings
DHCP Server
Firewall
NAT
Authorized IPs
Serial Settings
Serial Port 1
Serial Port 2
SSL Certificates
External devices
Logger config.
Modbus devices
Generic serial
Temperature sen.
IEC102 meter
W-MBus
GPS Receiver
Other
DynDNS
Private DynDNS
SMS Control
Periodic autoreset
Time Servers
Remote Console
Snmp
Tacacs+
Mqtt
Http / Https
CA-Certificates
Email config
Modbus Slave
Titan Scripts
Connectivity tools
Ditital I/O
Custom Skin
Led Config
Syslog
Backup / Factory
Firmware Upgrade
Reboot
SAVE CONFIG

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.7.14- Other → Passwords

There are three usernames to access the device configuration. The "admin" user, from which you will have access to the entire device configuration settings, and the "user" user, from which you will have access to the configurations you select (very useful if you personalise the device with your logos). In this section you can change the Password of both users. There is also a “guest” user, which will allow you to access the same menus as the “user” user, but without being able to make configuration changes.



Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.7.15- Other → CA Certificates

All services using secure connections under SSL/TLS connected to HTTPS or MQTTS servers must have the CA Root certificate used by that server, in order to verify the server's certificate. In this section, you can enter up to 2 custom CA Root servers (in PEM format).

It is also possible to select the "Allow all certificates" option. This is not recommended unless you have a secure connection (IPSec, OpenVPN, SIM with private APN, etc.), as the remote server will not be verified.

Additional Notes.

- The TITAN-based device has a built-in list of the most commonly used CA Root certificates. If you need to use a certificate that is unavailable, you can enter it here.

4.7.16- Other → Email Configuration

TITAN-based devices allow email notifications to be sent. They also allow emails to be sent via AT commands (e.g.: if you have a PLC and you want to send an SMS message or an EMAIL, you can do so by sending an AT command to the TITAN-based device).

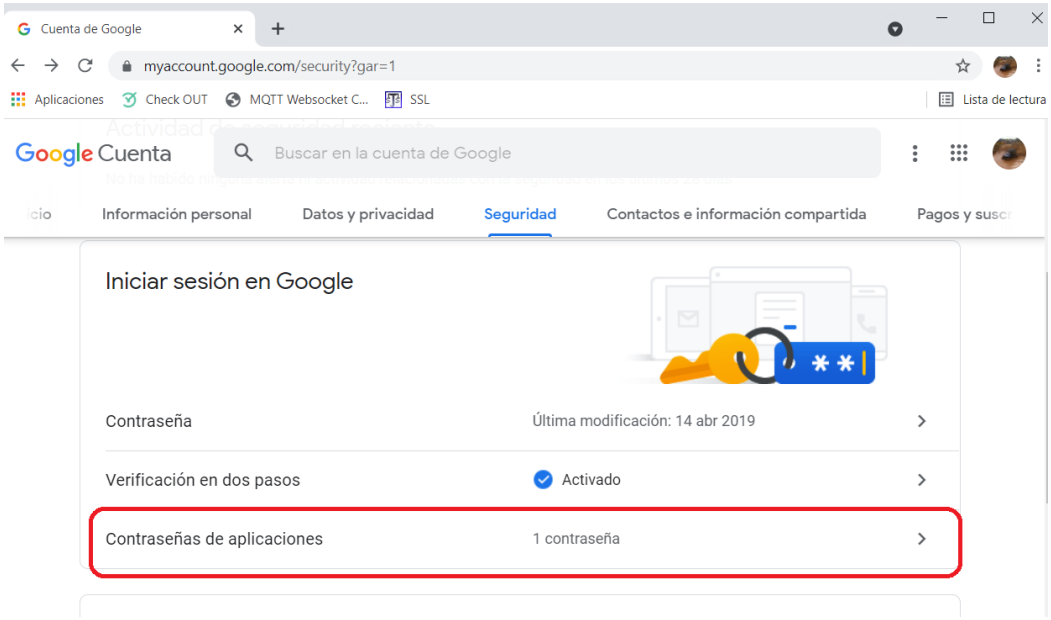
The TITAN-based device must be pre-configured to be able to send emails. That is, you must configure the SMTP server to be used in this section.

- ***Enabled:*** enables the email service.
- ***SMTP Server***: indicates the IP or DNS address of the SMTP server (e.g.: smtp.gmail.com, smtpout.secureserver.net, etc.)
- ***SMTP Port:*** TCP port of the SMTP server
- ***Authentication***: select whether the SMTP server requires authentication (usually yes)
- ***TLS:*** select whether the SMTP server requires TLS
- ***User:*** username of the source email account
- ***Password:*** Password of the source email account
- ***Email address:*** Source email address from which the emails will be sent

Additional Notes.

- Once the configuration process is complete, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

- Please note that if you use the GMAIL SMTP server, you will need to create an "application password". To do so, go to your "Google Account Manager", select the "Security" menu and add an "App Password".



4.7.17- Other → Modbus Slave

TITAN-based devices can be configured to act as a Modbus TCP Slave device and/or as a Modbus RTU Slave. For example, using the Modbus protocol, you can remotely switch (via 4G/3G/2G, Ethernet or Wifi) the digital outputs of the device, view the status of digital inputs, send SMSs, receive SMSs, or even send emails via Modbus.

The memory table with the Modbus addresses of the TITAN-based device registers is then available. The Modbus commands supported are: 0x03 to perform a reading and 0x10 to write.

Registration ID	R/W	Possible values:	Description
1	R	0 … 32635	Firmware version
2	R	0 … 32635	Firmware subversion
6	R	0 … 31	RSSI of the GSM signal
7	R	2,3,4	Technology used (2=2G/3=3G/4=4G)
8	R	1 … 31	Day
9	R	1 … 12	Month
10	R	2000 …. 2099	Year
11	R	0 … 23	Time
12	R	0 … 59	Minutes
13	R	0 … 59	Seconds
20	W	1	Performs a reset
30	RW	0.1	GPIO 0 (if any)
31	RW	0.1	GPIO 1 (if any)
32	RW	0.1	GPIO 2 (if any)
33	RW	0.1	GPIO 3 (if any)
34	RW	0.1	GPIO 4 (if any)
98	W	0 … 32635	Length of the AT command to be executed
99	R	0 … 32635	Length of the AT command response

100 … 354	W	ASCII	AT command text (in ASCII)
500 … 754	R	ASCII	Text of the AT command response (in ASCII)
1000	R/W	0, 1	1 indicates that a new SMS has been received. A PLC, after reading the SMS, must write a 0.
1001	R	0 … 18	Length of the telephone number to which the SMS has been sent
1002 … 1019	R	ASCII	Telephone number to which the SMS has been sent
1020	R	0 … 160	Length of SMS message text
1021 … 1180	R	ASCII	SMS text message
1021 … 1180	R	ASCII	SMS text message
10000	RW	0 … 65535	User registration 10000
10001… 19998	RW	ASCII	User registration from 10001 to 19998
19999	RW	0 … 65535	User registration 19999

AT commands via the Modbus protocol

Another very interesting feature of TITAN-based devices is the execution of AT commands via the Modbus protocol. For example, if you have a PLC that acts as a Modbus TCP Master, you can send a command to the device to execute an AT command via the Modbus TCP protocol, with the possibility to obtain data such as coverage, send an SMS, obtain the time, re-set the device, etc.

We will describe how to execute an AT command via Modbus below

Procedure for executing an AT command via Modbus

The procedure for executing an AT command via Modbus is very simple. It's best to explain this process using an example. Imagine you want to execute the AT+CSQ command to find out the coverage

1.- You will write the AT command in ASCII, starting from register 100.

Registration ID	Value	Description

100	65	ASCII of the letter: A
101	84	ASCII of the letter: T
102	43	ASCII of the character: +
103	67	ASCII of the letter: C
104	83	ASCII of the letter: S
105	81	ASCII of the letter: Q

2.- The AT+CSQ command is 6 characters long, so in order to execute the AT command, you will need to write to a 6 in register 98. After writing in register 98, the AT command will be executed immediately.

Registration ID	Value	Description
98	6	Length of the command to be executed

3.- Check the AT command execution result by reading register 99. This value read will show the length of the response. A value of 0 indicates that there has not yet been a response (execution has not finished, which will usually be less than 1 second). A value of >0 will show the length of the response.

Registration ID	Value	Description
99	28	Length of the command response

4.- Read the 28 registers where the response is contained from register 500 onwards

Registration ID	Value	Description
500	65	A
501	84	T

502	43	+
503	67	C
504	83	S
505	81	Q
506	13	\r
507	13	\r
508	10	\n
509	43	+
510	67	C
511	83	S
512	81	Q
513	58	:
514	32	[space]
515	**49**	**1**
516	**54**	**6**
517	44	,
518	57	9
519	57	9
520	13	\r
521	10	\n
522	13	\r
523	10	\n
524	79	O
525	75	K
526	13	\r
527	10	\n

In the case of this AT command, the relevant registers are 515 and 516, which indicate a coverage value of "16".

P.S. Remember that if you wish to use this feature to send an SMS, you must use the AT command indicated in this guide: AT^MTXTUNNEL=SMS,telephone,message

Procedures for receiving an SMS

The procedure for receiving SMS messages via Modbus TCP is very simple. Your PLC must periodically check register 1000. If the value read is a "1", it means that a new SMS message has arrived. You must then read it using the registers involved (1001-1180). Once read, you must write a “0” in register 1000.

Additional Notes.

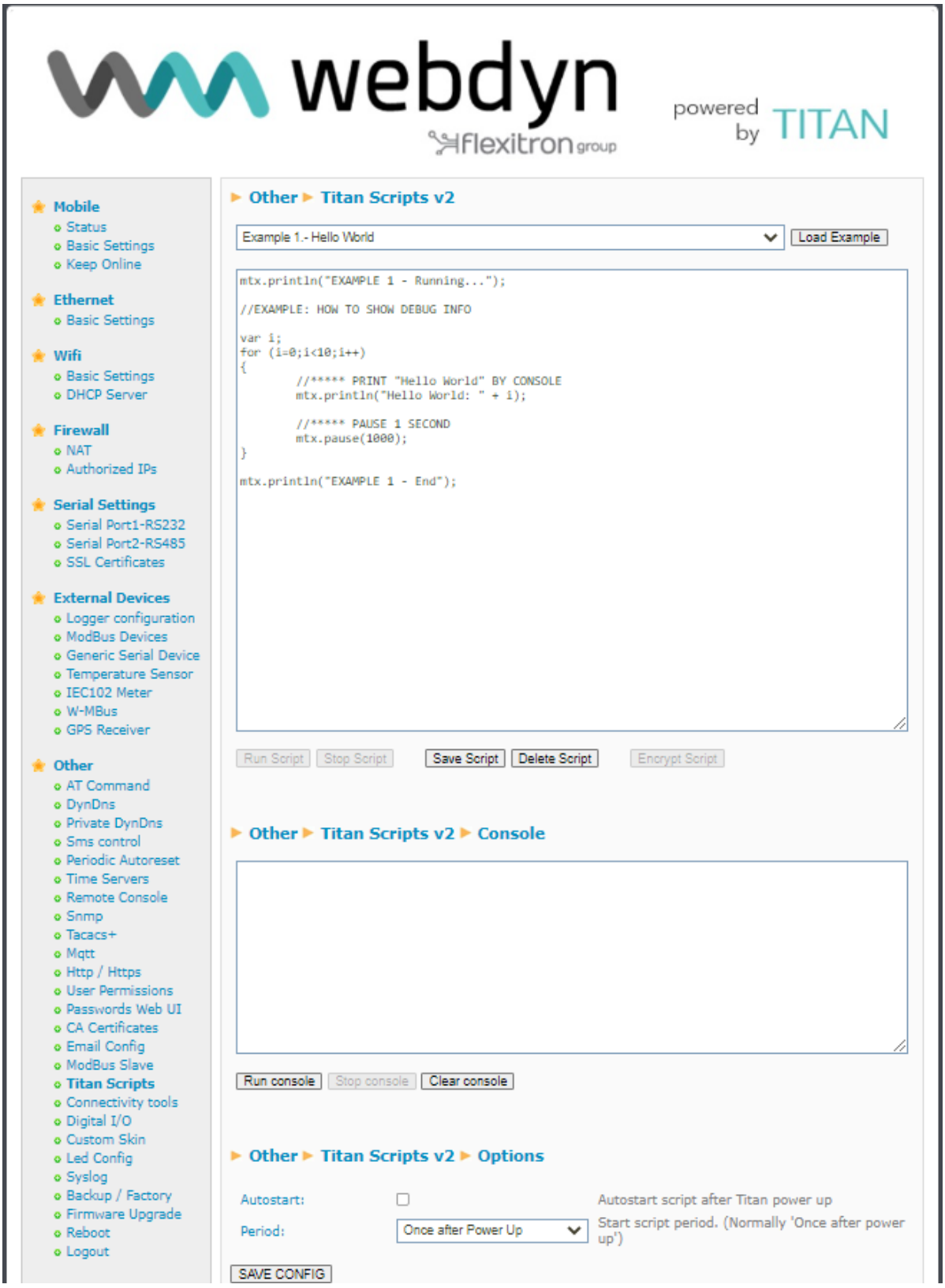
- Once the configuration process is complete, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

- See the Modbus registers table in the configuration menus of the TITAN-based device itself, in the Other > Modbus Slave section

4.7.18 - Other → TITAN Scripts

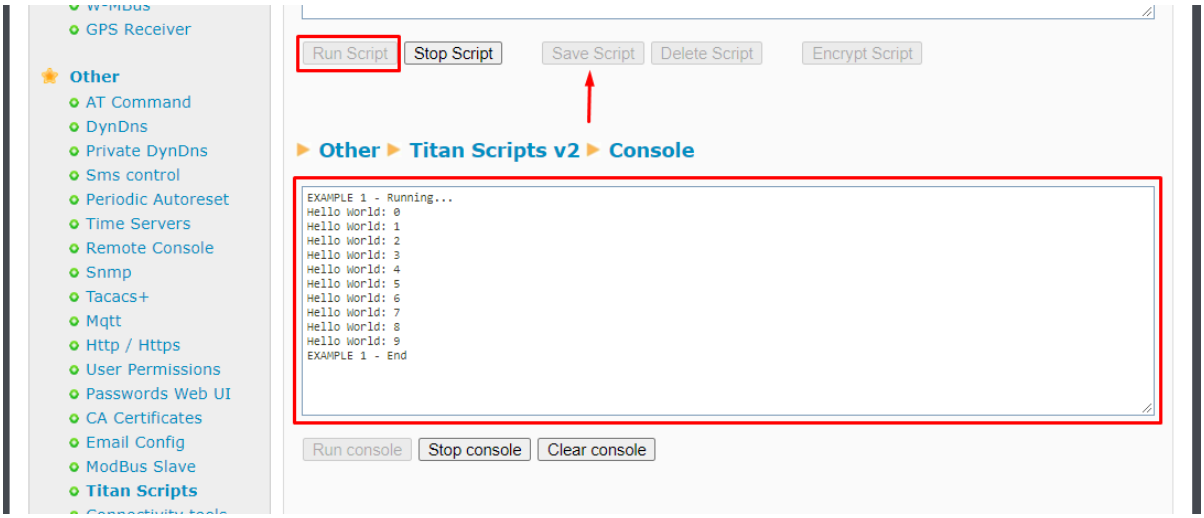
TITAN-based devices can be used to execute a user-programmable script in **JAVASCRIPT (ECMAScript 5)**
https://www.w3schools.com/jsref/default.asphttps://www.w3schools.com/jsref/default.asp From this script you can access various features of the TITAN-based device to make small customisations (like reading devices through the serial port, reading digital inputs, changing a digital output, reading a Modbus device, sending an SMS message, sending an EMAIL, an MQTT message to a certain topic, sending a SNMP TRAP, saving and reading files on the device, etc. (This is easier to understand through the various examples of the TITAN-based devices).

The TITAN Scripts configuration display looks like this:

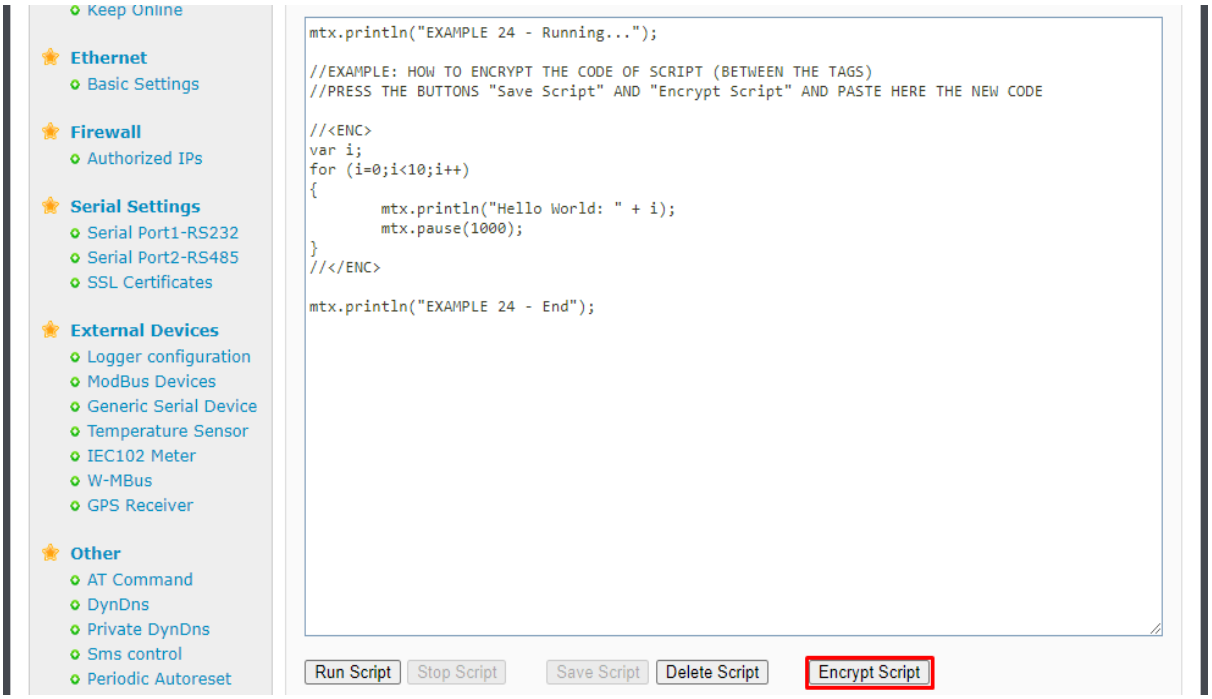


In the top drop-down menu, you have multiple examples of all the features you can use. To load an example, select the example of your choice from the drop-down menu (the "Hello World" example is recommended for the first time) and then click on the "Load Example" button. The source code of the example will then appear on the screen.

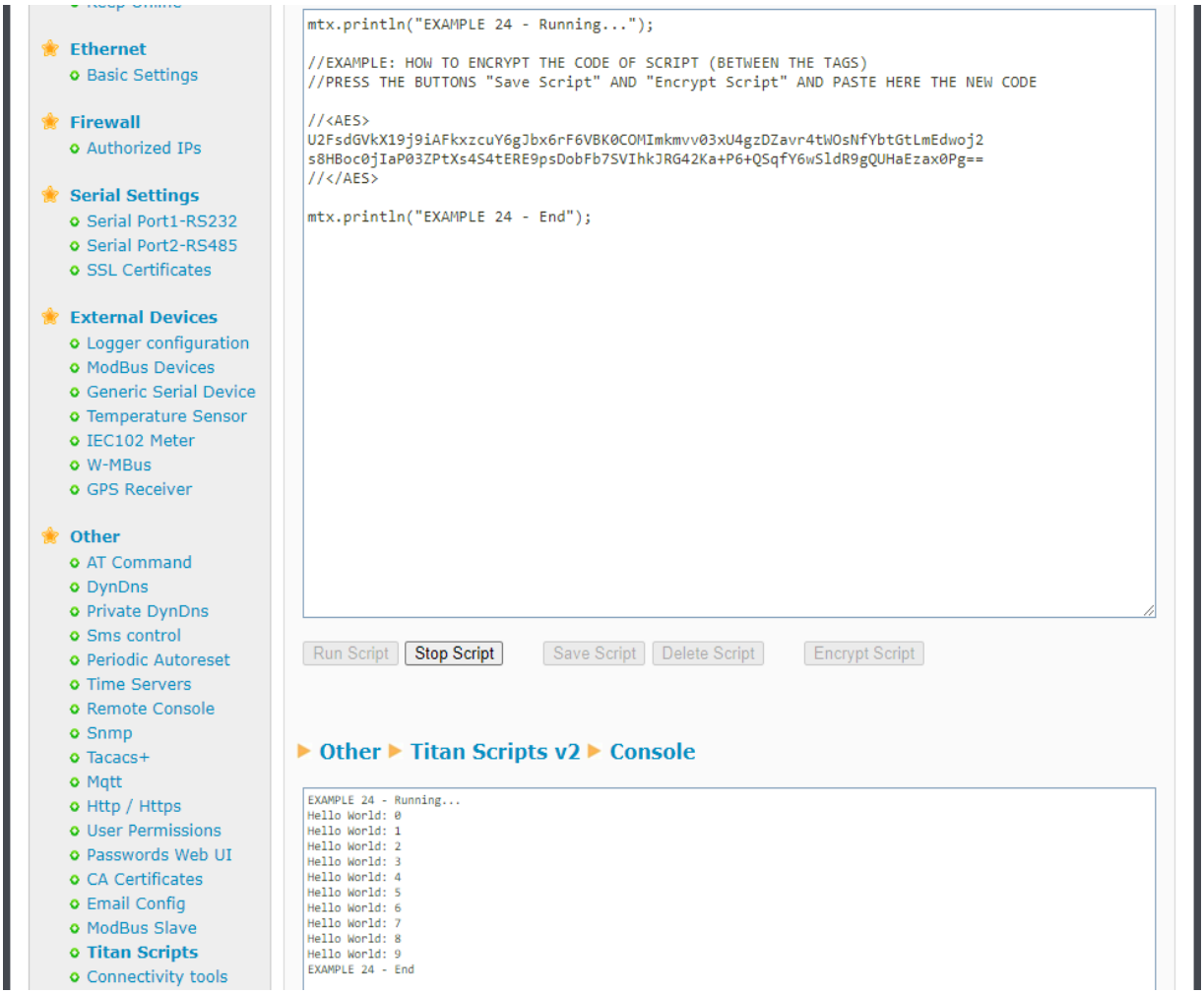
Now, click the "Save Script" button to save the example code in the execution memory of the TITAN-based device. You can then click on the "Run Script" button to run the script, and you will be able to see the elements of the "mtx.println()" program within a few seconds in the console section.



If you wish to encrypt the script code, click on the "Encrypt Script" button and a pop-up window with a new code will be generated.



After pasting this new code into the script, the TITAN-based device will be able to run it in the same way.



For more information on script encryption, see examples 24 and 25.

TITAN Script Configuration Options.

Once your script is developed and running, you will want it to run autonomously every time the TITAN-based computer restarts. To do so, you must check the "Autostart" box. Once this box is checked, you must specify the autostart mode. The usual way will be "Once after Power Up", i.e.: the script will be executed once the TITAN-based device has started up. This is optimal for most applications of this type:

```
while (true) {

…

}
```

in which the code is always executed cyclically. However, it might be more convenient to have the script run automatically at certain time intervals (every minute, every 2 minutes, 5 minutes, etc.) to perform certain tasks. For example, imagine a case where we need a script to read a Modbus RTU device every 15 minutes and send the data to an MQTT server. In this case you may find it easier to run the script every 15 minutes.

The TITANScripts "mtx" object.

As indicated in the previous paragraphs, TITANScripts allow you to program small scripts in JavaScript (ECMAScript 5) to interact with certain features of TITAN-based devices. These features are accessed through certain functions of an "mtx" object. We will now describe each one. It will be easier to understand this if you study the short examples included in the section "Other → TITAN Scripts". **Please bear in mind that TITAN Scripts have been designed to carry out small customisations in the behaviour of TITAN-based devices, and not to program any kind of application, since this could be done in other languages such as "Java", "C++", "Python", etc. Basically, everything you can do is included in the examples.**

Functions related to IOx DIGITAL INPUTS / OUTPUTS:

In the "Notes" section of each function, you will find a list of example scripts available using that function. These examples are hosted within the router itself via the "Other TITAN Scripts" menu.

- **Function: mtx.ioGet(idIO)**

  Description: allows you to obtain the status of a digital input/output.
  Parameters: idIO: int 0,1,2,3,4, etc. (IO identifier)
  Response: int (0 = input not enabled, 1 = input enabled, -1 = error)
  NB: see examples: 3, 15 and 16

- **Function: mtx.ioSet(idIO,"VALUE)**

  Description: allows you to change the status of a digital input/output.
  Parameters: idIO: (int) 0,1,2,3,4, etc. (IO identifier)
  value: (int) 0,1 (logic value of the digital output)
  Response: boolean (true=operation OK, false=operation ERROR)
  Notes: see examples: 16

- **Function: mtx.counterGet(idIO)**
  Description: allows you to obtain the value of the pulse meter linked to an input.
  Parameters: idIO: int 0,1,2,3,4, etc. (digital input identifier)

Response: long meter value
Notes: see examples: 34

- **Function: mtx.counterSet(idIO, value)**
  Description: allows you to change the initial value of a pulse meter
  Parameters: idIO: (int) 0,1,2,3,4, etc. (IO identifier)
  value: (long) (meter value)
  Response: boolean (true=operation OK, false=operation ERROR)
  Notes: see examples: 34

Functions related to the router LEDs:

- **Function: mtx.ledSet(idLed, value)**
  Description: Turns on/off an LED
  Parameters: idLed: (int) 0,1,2, etc. (device LED identifier)
  value: (int) 0 = off, 1 = on, 2 = slow flashing, 3 = fast flashing.
  Response: boolean (true = operation OK, false = operation ERROR)
  NB: see examples: 21

Functions related to AT COMMANDS:

- **Function: mtx.atSend(command, timeout)**
  Description: allows you to execute an AT command on the TITAN-based equipment itself
  Parameters: command: (String) AT command to execute
  timeout: (int) milliseconds of timeout
  Response: String: response to AT command. “”= error
  Notes: see examples: 2

MQTT-related functions:

- **Funtion:** **mtx.mqttIsConnected()**
  Description: returns if the TITAN-based device is connected to an MQTT broker
  Parameters: None.
  Response: boolean (true = Yes, false = No)
  NB: see examples: 4, 35

- **Function: mtx. mqttSend(message, topic, qos)**
  Description: allows you to send a text MQTT message to a given mqtt topic with a given QoS
  Parameters: message: (String) Text message to be sent
  topic: (String) Topic MQTT to send the message to
  qos: (int) Submission QoS (0,1,2)
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 4, 8 and 35

- **Function: mtx. mqttSendArray(message, topic, qos)**
  Description: allows you to send a byte-array MQTT message to a given mqtt topic with a given QoS
  Parameters: message: (byte array) Text message to be sent
  topic: (String) Topic MQTT to send the message to
  qos: (int) Submission QoS (0,1,2)
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 35

- **Function: mtx. mqttSubscribe(topic)**
  Description: TITAN scripts enable subscription to 1 MQTT topic in order to receive communications from a broker. You can subscribe to this topic via this function.
  Parameters: topic: (String) Topic MQTT where the TITAN scripts will receive messages
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 35

- **Function:** **mtx. mqttGetArray()**
  Description: function to check whether data has arrived at the configured topic using the aforementioned function mtx.mqttSubscribe (topic)
  Parameters: None
  Response: byte array: (Null = no data, !null = byte array with data)
  NB: see examples: 35

Functions related to TRAPS SNMP:

- **Function:** **mtx.trapSend(oid, message, severity)**
  Description: sends an SNMP trap
  Parameters: oid: (String) OID of the SNMP trap
  message: (String) Text associated to the SNMP trap
  severity (int) Trap severity (1, ... ,7)
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 5, 9

SMS-related functions:

- **Function:** **mtx.smsSend(phoneNumber, textMessage)**
  Description: sends an SMS message to a phone number
  Parameters: phoneNumber: (String) telephone number receiving message
  textMessage: (String) text in SMS message
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 3, 10, 17, 18 and 40

- **Function:** **mtx.smsRead()**
  Description: checks whether a new SMS message has arrived
  Parameters: None
  Response: array String: (null=no new SMS, data[0]=telephone number
  data[1]=SMS message)
  Notes: see examples: 40

HTTP-related functions:

- **Function:** **mtx.httpRequest(url, headers,method,data,dataType)**
  Description: sends an email to an email address
  Parameters: url: (String) connection
  Url headers: (HashMap) custom HTTP headers
  method: (String) GET or POST
  data: (String) data to be sent with the http request
  dataType: (String) data type to be sent with the http request
  NB: see examples: 39

EMAILS-related functions:

- **Function:** **mtx.emailSend(address, subject)**
  Description: sends an email to an email address
  Parameters: address: (String) email destination address
  subject: (String) subject of sent email
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 6, 11

TELEGRAM-related functions:

- **Function:** **mtx. telegramSendTextMessage(apiToken, chatID,message) Description:**
  sends a text message to a Telegram group chat
  Parameters: **apiToken:** (String) Telegram bot API token
  chatID: (String) chat identifier to which the message will be sent
  message: (String) text message to be sent
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 41, 42

SIM CARD-related functions:

- **Function:** **mtx.simID()**
  Description: returns the SIM card that is being used
  Parameters: None
  Response: int: (-1=ERROR, 0=no SIM, 1=SIM1, 2=SIM2)
  Notes: see examples: 18

- **Function:** **mtx.simInserted()**
  Description: **returns if the current SIM card is inserted**
  Parameters: None
  Response: int: (-1=ERROR, 0=no insertada, 1=insertada)
  Notes: see examples: ---

- **Function:** **mtx.simStatus()**
  Description: **returns the current SIM status**
  Parameters: None

Response: int: (-1=ERROR, 0=ERROR de SIM, 1=SIM OK)
Notes: see examples: 18

Functions related to SERIAL PORTS:

- **Function:** **mtx.serialOpen(idPort,baudrate,databits,parity,stopbits,flowcontrol)**
  Description: Opens a serial port
  Parameters: idPort: (int) serial port ID (1,2)
  baudrate: (int) speed (460800, 230400, ... , 1200, 600, 300)
  databits: (int) number of bits per data (8,7)
  parity: (int) 0 = none, 1 = odd, 2 = even
  stopbits: (int) 1.2
  flowcontrol: (int) 0 = no, 1 = hardware (if available)
  Response: boolean: true: OK, false: ERROR
  Notes: see examples: 19

- **Function:** **mtx.serialClose(idPort)**
  Description: closes a serial port
  Parameters: idPort: (int) serial port ID (1,2)
  Response: boolean: true: OK, false: ERROR
  Notes: see examples: ---

- **Function:** **mtx.serialRead(idPort)**
  Description: reads and returns data read by an RS232 or RS485 serial port
  Parameters: **idPort:** (int) serial port id (1,2)
  Response: byte []: byte array with read data
  null: no data read.
  NB: see examples: 19

- **Function:** **mtx.serialWrite(idPort,data,start,length)**
  Description: send a byte array via the serial port
  Parameters: idPort: (int) serial port ID (1,2)
  data: (byte[]) byte array for e
  start: (int) start index for sending of byte array
  length: (int) number of bytes to be sent.
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 19

MODBUS-related functions:

- **Function:** **mtx.modbusRTUGetWords(address, command, start, numRegisters)**
  Description: reads and returns Modbus registers from a modbus RTU device
  Parameters: address: (int) Modbus RTU address (1 ... 254)
  command: (int) Modbus command 3 or 4
  start: (int) initial register address to be read
  numRegisters: (int) number of registers to read (1 ... 64)
  Response: int []: WORDS array with the data read (values 0 ... 65535)
  null: Reading ERROR.
  NB: see examples: 7, 8, 9, 10, 11, 12, 22, 30

- **Function:** **mtx.modbusRTUSetWords(address, command, start, values[])**
  Description: writes Modbus registers to a modbus RTU device
  Parameters: address: (int) Modbus RTU address (1 ... 254)
  command: (int) Modbus command 16 or 6
  start: (int) initial register address to be written
  values[]: (int[]) array with the WORDS to be written
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 12

- **Function:** **mtx.modbusRTUGetBits(address, command, start, numBits) Description:**
  reads and returns the bit-type Modbus registers of a modbus RTU device
  Parameters: address: (int) Modbus RTU address (1 ... 254)
  command: (int) Modbus command 1 or 2
  start: (int) address of initial bit register to be read (1 ... 65535)
  numBits: (int) number of bits to read (1 ... 64)
  Response: int []: BITS array with the bits read (values 0,1)
  null: Reading ERROR
  Notes: see examples: 13

- **Function:** **mtx.modbusRTUSetBit(address, command, start, value)**
  Description: writes 1 bit-type Modbus register to a modbus RTU device
  Parameters: address: (int) Modbus RTU address (1 ... 254)
  command: (int) Modbus command (5)
  start: (int) initial bit register address to be written
  values: (int) value to be written (0,1)
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 14

- **Function:** **mtx.modbusTCPGetWords(ip, port, command, start, numRegisters)**
  Description: reads and returns the Modbus registers of a Modbus TCP device
  Parameters: ip: (String) IP address of the Modbus TCP device
  port: (int) TCP port of the Modbus TCP device
  command: (int) Modbus command 3 or 4
  start: (int) initial register address to be read
  numRegisters: (int) number of registers to read (1 ... 64)
  Response: int []: WORDS array with the data read (values 0 ... 65535)
  null: Reading ERROR.
  NB: see examples: ---

- **Function:** **mtx.modbusTCPSetWords(ip, port, command, start, values[])**
  Description: write Modbus registers to a Modbus TCP device
  Parameters: ip: (String) IP address of the Modbus TCP device
  port: (int) TCP port of the Modbus TCP device
  command: (int) Modbus command 16 or 6
  start: (int) initial register address to be written
  values[]: (int[]) array with the WORDS to be written
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 22

- **Function:** **mtx.modbusTCPGetBits(ip, port, command, start, numBits) Description:**
  reads and returns the Modbus bit-type registers of a Modbus TCP device
  Parameters: ip: (String) IP address of the Modbus TCP device
  port: (int) TCP port of the Modbus TCP device
  command: (int) Modbus command 1 or 2
  start: (int) address of initial bit register to be read (1 ... 65535)
  numBits: (int) number of bits to read (1 ... 64)
  Response: int []: BITS array with the bits read (values 0,1)
  null: Reading ERROR.
  NB: see examples: ---

- **Function:** **mtx.modbusTCPSetBit(ip, port, command, start, value)**
  Description: writes 1 Modbus bit-type register to a Modbus TCP device
  Parameters: ip: (String) IP address of the Modbus TCP device
  port: (int) TCP port of the Modbus TCP device
  command: (int) Modbus command (5)
  start: (int) initial bit register address to be written
  values: (int) value to be written (0,1)

Response: boolean: (true=operation OK, false=operation ERROR)
Notes: see examples: ---

- **Function:** **mtx.modbusSaveWord(idRegister, registerValue)**
  Description: writes a word-type value to the user memory area
  Parameters: idRegister: (int) register address (10000 ... 19,999)
  registerValue: (int) register value (0 ... 65,535)
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: ---

- **Function:** **mtx.modbusSaveDWord(idRegister, registerValue)**
  Description: writes a double word-type value to the user memory area
  Parameters: idRegister: (int) register address (10000 ... 19,999)
  registerValue: (long) register value (0 ... 4294967295)
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 49

- **Function:** **mtx.modbusReadWord(idRegister)**
  Description: reads a word-type value from the user memory area
  Parameters: idRegister: (int) register address (10000 ... 19999)
  Response: int: value of register read (0 ... 65,535) -1: error
  Notes: see examples: ---

- **Function:** **mtx.modbusReadDWord(idRegister)**
  Description: reads a double word-type value from the user memory area
  Parameters: idRegister: (int) register address (10000 ... 19999)
  Response: long: value of register read (0 ... 4294967295) -1: error
  Notes: see examples: ---

IEC-102 meter-related functions:

- **Function:** **mtx.iec102AddMeter(idMeter,linkAddress,measAddress,password)**
  Description: Adds a meter to the meter manager. Maximum 100.
  Parameters: idMeter (String) meter identifier
  linkAddress (int) meter link address
  measAddress (int) meter measuring point address
  password (long) meter password

Response: boolean true: meter added correctly.
false: error
Notes: see examples: 32, 47, 48, 49

- **Function:** **mtx.iec102DeleteAllMeters()**
  Description: removes all meters from the meter manager.
  Parameters:
  Response: boolean true: meters removed correctly.
  false: error
  Notes: see examples: 32, 47, 48, 49

- **Function:** **mtx.iec102GetAllMeters()**
  Description: returns a list of all meters stored in the meter manager.
  Parameters:
  Response: ArrayList [] List of all meter identifiers
  Notes: see examples: 48, 49

- **Function:** **mtx.iec102GetPresence()**
  Description: Checks for the presence of a meter
  Parameters: idMeter (String) meter identifier
  Response: int - 2: not yet initialised
  -1: bus busy with reading
  0: meter detected
  >0: error
  Notes: see examples: 47

- **Function:** **mtx.iec102SetInstantValues (idMeter)**
  Description: executes the command to read the instantaneous meter values.
  Parameters: idMeter (String) meter identifier
  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: 32, 48 and 69

- **Function:** **mtx.iec102GetInstantValues(idMeter)**
  Description: reads the instantaneous meter values
  Parameters: idMeter (String) meter identifier
  Response: JSON string with the data read from the meter "" ERROR
  Notes: see examples: 32, 48 and 69

- **Function:** **mtx.iec102SetFiscalClose(idMeter, hourIni, minuteIni, dayIni, monthIni, yearIni, hourEnd, minuteEnd, dayEnd, monthEnd, yearEnd)**
  Description: executes the command to read the fiscal close values
  Parameters: idMeter (String) meter identifier
  hourIni (int) initial hour (0 ... 23)
  minuteIni (int) initial minute (0 ... 59)
  dayIni (int) initial day (1 ... 31)
  monthIni (int) initial month (1 ... 12)
  yearIni (int) initial year (22 .. 99)
  hourEnd (int) end hour (0 ... 23)
  minuteEnd (int) end minute (0 ... 59)
  dayEnd (int) end day (1 ... 31)
  monthEnd (int) end month (1 ... 12)
  yearEnd (int) end year (22 ... 99)

  Response: boolean: (true=operation OK, false=operation ERROR)
  Notes: see examples: ---

- **Function:** **mtx.iec102GetFiscalClose(idMeter)**
  Description: reads the meter closing values
  Parameters: idMeter (String) meter identifier
  Response: JSON string with the data read from the meter "" ERROR
  Notes: see examples: ---

- **Function:** **mtx.iec102SetIntegratedTotal(idMeter, hourIni, minuteIni, dayIni, monthIni, monthIni, yearIni, hourEnd, minuteEnd, dayEnd, monthEnd, yearEnd, mode, period) Description:** executes the command to read the integrated totals (load curve)
  Parameters: idMeter (String) meter identifier
  hourIni (int) initial hour (0 ... 23)
  minuteIni (int) initial minute (0 ... 59)
  dayIni (int) initial day (1 ... 31)
  monthIni (int) initial month (1 ... 12)
  yearIni (int) initial year (22 ... 99)
  hourEnd (int) final hour (0 .. 23)
  minuteEnd (int) final minute (0 ... 59)
  dayEnd (int) final day (1 ... 31)
  monthEnd (int) final month (1 ... 12)
  yearEnd (int) final year (22 ... 99)
  mode (int) 0 = absolute, 1 = incremental
  period (int) 0 = hourly, 1 = every quarter of an hour

Response: boolean: (true=operation OK, false=operation ERROR)
Notes: see examples: 52

- **Function:** **mtx.iec102GetIntegratedTotal(idMeter)**
  Description: reads the integrated total values (load curve) from the meter
  Parameters: idMeter (String) meter identifier
  Response: JSON string with the data read from the meter "" ERROR
  Notes: see examples: 52

GPS-related functions:

- **Function:** **mtx.gpsGetData()**
  Description: returns the actual GPS position of the TITAN-based device.
  Parameters: None
  Response: JSON string with GPS data "" ERROR (if no NMEA frames are detected)
  Notes: see examples: 33

LOGGER-related functions:

- **Function:** **mtx.loggerWrite(data)**
  Description: writes data to the internal TITAN-based device logger
  Parameters: data: (String) data to be stored in the logger
  Response: boolean true = successful operation
  false = error (e.g. data size greater than logger register size).
  NB: see examples: 26

FILE-related functions:

- **Function:** **mtx.fileWrite(index, data, append)**
  Description: writes data to a file within the TITAN-based device
  Parameters: index: (int) file index (0 ... 9)
  data: (String) data to be written to the file
  append: (boolean) true=adds the text to the existing file
  false=deletes the file and saves the text
  Response: boolean true = successful operation
  false = error (e.g. file data size greater than 100KB).
  NB: see examples: 27, 28 and 30

- **Function:** **mtx.fileRead(index)**
  Description: reads data from a file in TITAN-based device
  Parameters: index: (int) file index (0 ... 9)
  Response: String data contained in the file
  NB: see examples: 27, 29

- **Function:** **mtx.fileSize(index)**
  description: returns the data size of a file
  Parameter(s): index: (int) file index (0 ... 9)
  Response: long file size (in bytes)
  Notes: see examples: 28, 30

- **Function:** **mtx.fileFtpDownload(tls, index, server, port, username, pass, remoteFile, timeout, addExtension)**
  Description: receives a TITAN data file from a remote server via FTP
  Parameters: tls: (int) 0 = FTP, 1 = FTPS
  index: (int) index of the file (0 ... 9) where to store the data
  server: (String) IP or DNS of the remote FTP server
  port: (int) 0 … 65535
  username: (String) FTP server username
  pass: (String) FTP server password
  remoteFile: (String) Remote file to be downloaded

	timeout:	(int) seconds of timeout to complete downloading
	addExtension:	(String) If the file is downloaded, this extension is added
Response:	boolean	true = successful operation
		false = error
Notes:	see examples: 29	

- **Function:** **mtx.fileFtpUpload(tls, index, server, port, username, pass, remoteFile, timeout)**

Description:	sends a TITAN data file to a remote server via FTP	
Parameters:	tls:	(int) 0 = FTP, 1 = FTPS
	index:	(int) file index (0 ... 9)
	Server:	(String) IP or DNS of the remote FTP server
	port:	(int) 0 … 65535
	username:	(String) FTP server username
	pass:	(String) FTP server password
	remoteFile:	(String) Remote file to be downloaded
	timeout:	(int) Seconds of timeout to complete sending
Response:	boolean	true = successful operation
		false = error
Notes:	see examples: 28, 30	

- **Function:** **mtx.fileFtpRename(tls, server, port, username, pass, currentFileName, newFileName, timeout)**

Description:	sends a TITAN data file to a remote server via FTP	
Parameters:	tls:	(int) 0 = FTP, 1 = FTPS
	server:	(String) IP or DNS of the remote FTP server
	port:	(int) 0 … 65535
	username:	(String) FTP server username
	pass:	(String) FTP server password
	currentFileName	(String) Current remote file name
	newFileName	(String) New remote file name
	timeout:	(int) Seconds of timeout to complete the operation
Response:	boolean	true = successful operation
		false = error
Notes:	see examples: ---	

- **Function:** **mtx.scriptFtpDownload(tls, server, port, username, pass, remoteFile, timeout, addExtension)**

Description:	downloads the script file from an FTP server

Parameters: tls: (int) 0 = FTP, 1 = FTPS
server: (String) IP or DNS of the remote FTP server
port: (int) 0 … 65535
username: (String) FTP server username
pass: (String) FTP server password
remoteFile: (String) remote script to be downloaded
timeout: (int) seconds of timeout to complete downloading
addExtension: (String) If the file is downloaded, this extension is added
Response: boolean true = successful operation
false = error
Notes: see examples: 31

MODEM-related functions:

- **Function:** **mtx.imeiCheck(imei, message)**
  Description: checks if the IMEI set as a parameter is correct
  Parameters: imei: (String) IMEI to be checked
  message: (String) message to return if incorrect
  Response: void:
  Notes: see examples: 25

- **Function:** **mtx.imeiGet()**
  Description: **returns the IMEI of the internal modem of the TITAN-based device**
  Parameters: None
  Response: String The IMEI of the internal modem of the TITAN-based device
  NB: see examples: 30

OTAP-related functions:

- **Function:** **mtx.otapFTPConfig(tls, server, port, user, pass, remoteConfigFile,secTimeout,addExtension);**
  Description: executes a remote configuration download command via FTP/FTPS.
  If the configuration is downloaded successfully, it is possible to rename the remote file (so that it is not downloaded again) and install the configuration.
  Parameters: tls: (int) 0=no TLS, 1= TLS

server: (String) FTP server
port: (int) FTP port
user: (String) FTP username
pass: (String) FTP password
remoteConfigFile: (String) remote file (e.g.: path1/path2/file)
sectimeout: (int) seconds of timeout
addExtension: (String) If downloaded, this extension is added to the remote file (so that it is not downloaded again)

Response: boolean true=process started, false; process not started

Notes: see examples: 37

- **Function:** **mtx.otapFTPLastStatus()**
  Description: returns the current status of an OTAP process
  Parameters:
  Response: int 0 = none, 1 = running, 2 = OK, 3 = ERROR **NB**: see examples: 37

WATCHDOG-related functions:

- **Function:** **mtx.watchdogSet(mode, timeout)**
  Description: starts the watchdog
  Parameters: int mode: (int) 0=reset mode
  timeout: (int) seconds to re-set if not refreshed (0, 30 ... 100000). 0 = Disabled.
  Response: boolean true = OK, false: ERROR
  Notes: see examples: 20

- **Function:** **mtx.watchdogKick()**
  Description: refreshes the watchdog so that it does not restart the modem
  Parameters: None
  Response: boolean true = OK, false: ERROR
  Notes: see examples: 20

Time-related functions:

- **Function:** **mtx.pause(mseconds)**
  Description: creates a pause the length of the configured milliseconds
  Parameters: seconds (long) milliseconds to wait
  Response: boolean true = successful operation
  false = error
  Notes: see examples: 1, 2, 3, 4, 5,…,, 35

- **Function:** **mtx.timeStamp()**
  Description: returns the date and time of the TITAN-based device
  Parameters: None
  Response: String date and time of the TITAN-based device
  NB: see examples: 28, 30

- **Function:** **mtx.timeGetYear()**
  Description: returns the year
  Parameters: None
  Response: int current year with four digits
  NB: see examples: ---

- **Function:** **mtx.timeGetMonth()**
  Description: returns the month
  Parameters: None
  Response: int current month (1 = January ... 12 = December)
  NB: see examples: ---

- **Function:** **mtx.timeGetDay()**
  Description: returns the day
  Parameters: None
  Response: int current day (1 ... 31)
  NB: see examples: ---

- **Function:** **mtx.timeGetHour()**
  Description: returns the hour
  Parameters: **None**
  Response: int current hour (0 ... 23)
  NB: see examples: ---

- **Function:** **mtx.timeGetMinute()**
  Description: returns the minutes
  Parameters: **None**

Response: int current minutes (0 ... 59)
NB: see examples: ---

- **Function:** **mtx.timeGetSecond()**
  Description: returns the seconds
  Parameters: **None**
  Response: int current seconds (0 ... 59)
  NB: see examples: ---

Functions related to sockets:

- **Function:** **mtx. socketOpen (idSocket,ssl,ip,port)**
  Description: opens and connects a TCP Client socket to an IP/DNS
  Parameters: idSocket (int) socket identifier (0 ... 9)
  ssl (int) 0=no, 1=yes
  idSocket (String) IP or DNS address of the remote server
  TCP port (int) TCP port of the remote server (0 ... 65,535)
  Response: boolean true = successful operation
  false = error
  Notes: see examples: 45, 46

- **Function:** **mtx. socketClose (idSocket)**
  Description: closes a TCP Client socket
  Parameters: idSocket (int) socket identifier (0 ... 9)
  Response: boolean true = successful operation
  false = error
  Notes: see examples: 45, 46

- **Function:** **mtx. socketIsOpen(idSocket)**
  Description: checks if a TCP Client socket has an established connection
  Parameters: idSocket (int) socket identifier (0 ... 9)
  Response: boolean true = yes, there is an established connection
  false = no, there isn't an established connection.
  NB: see examples: 45, 46

- **Function:** **mtx. socketSend(idSocket, data[])**
  Description: sends data over a TCP(IP) socket

Parameters: idSocket (int) socket identifier (0 ... 9)
values[]: (byte[]) byte array to be sent
Response: boolean: (true=operation OK, false=operation ERROR)
Notes: see examples: 45, 46

- **Function:** **mtx. socketIsData(idSocket)**
  Description: checks if there is pending data to read in a TCP Client socket
  Parameters: idSocket (int) socket identifier (0 ... 9)
  Response: boolean true = there is pending data
  false = there is no pending data
  Notes: see examples: 45, 46

- **Function:** **mtx. sockeRead(idSocket)**
  Description: reads the bytes available in a TCP Client socket
  Parameters: idSocket (int) socket identifier (0 ... 9)
  Response: byte[] Byte array with the received data
  null: error
  Notes: see examples: 45, 46

Data conversion functions:

- **Function:** **mtx.convert2WordToUnsignedInt32 (word1, word2)**
  Description: converts two words to 32-bit unsigned integer format
  Parameters: word1 (int) 0 ... 65535 word2 (int) 0 ... 65535
  Response: unsigned int32 0 ... 4294967295
  Notes: see examples: 14c

- **Function:** **mtx.convert2WordToSignedInt32 (word1, word2)**
  Description: converts two words to 32-bit signed integer format
  Parameters: word1 (int) 0 ... 65535 word2 (int) 0 ... 65535
  Response: signed int32 -2147483648 ... 2147483647
  Notes: see examples: 14c

- **Function:** **mtx.convert2WordToFloat (word1, word2)**
  Description: converts two words to float format
  Parameters: word1 (int) 0 .. . 65535 word2 (int) 0 ... 65535
  Response: Float -3.4028235x10^38 ... 3.4028235x10^38
  Notes: see examples: 14c

- **Function:** **mtx.convert4WordToSignedInt64 (word1, word2, word3, word4)**
  Description: converts four words to 64-bit signed integer format
  Parameters: word1 (int) 0 .. . 65535
  word2 (int) 0 ... 65535
  word3 (int) 0 ... 65535
  word4 (int) 0 ... 65535
  Response: String -9223372036854775808 ... 9223372036854775807
  Notes: see examples: 14C
  In value is returned in String format, because javascript does not support 64-bit integers. Useful for forming JSONs for submission to platforms.

- **Function:** **mtx.convert4WordToDouble (word1, word2, word3, word4)**
  Description: converts four words to 64-bit Double format
  Parameters: word1 (int) 0 ... 65535
  word2 (int) 0 ... 65535
  word3 (int) 0 ... 65535
  word4 (int) 0 ... 65535
  Response: Double -4.9x10^324 ... 1.7976931348623157x10^308
  Notes: see examples: 14c

VARIOUS functions:

- **Function:** **mtx.mobileGetIP()**
  Description: returns the IP address associated with the Mobile WAN
  Parameters: None
  Response: String “”= None, “x.x.x.x”
  NB: see examples: ---

- **Function:** **mtx.powerStatus()**
  Description: returns if the router has external power. Useful for devices with an internal battery or supercap.
  Parameters: None
  Response: int 0 = no, 1 = yes, -1 = error
  NB: see examples: 6

- **Function:** **mtx.println(data)**
  Description: sends text data to the debug console
  Parameters: data (String) text data to be written to console
  Response: void
  Notes: see examples: 1, 2, 3, 4, 5,…,, 35

- **Function:** **mtx.ping(ip, timeout)**
  Description: generates a ping to a remote IP or DNS
  Parameters: ip (String) Remote IP or DNS to execute ping
  timeout (int) milliseconds of timeout
  Response: boolean true = ping successful
  false = ping error
  Notes: see examples: 17

- **Function:** **mtx.base64Encode(data)**
  Description: encodes a byte array in base64
  Parameters: data (byyte[]) byte array to encode
  Response: String data encoded en base64
  “”= error
  Notes: see examples: 26

- **Function:** **mtx.stringToByteArray(data)**
  Description: converts a String to a byte array
  Parameters: data (String) data to be converted into a byte array
  Response: byte[] byte array
  null = error
  Notes: see examples: 51

- **Function:** **mtx.byteArrayToString (data, init, length)**
  Description: converts a byte array into a String
  Parameters: data (byte[]) byte array
  init (int) position
  length (int) number of bytes to be converted
  Response: String Converted string
  "" = error
  Notes: see examples: ---

Function: **mtx.configParamGet(paramName)**
Description: reads the value of a configuration parameter from the Titan-based device
Parameters: paramName (String) name of the configuration parameter

Response:	String	value of the configuration parameter.
NB:	see examples: ---	
Function:	**mtx.configParamSet(paramName,paramValue)**	
Description:	writes the value of a Titan configuration parameter	
Parameters:	paramName	(String) name of the configuration parameter
Parameters:	paramValue	(String) value of the configuration parameter
Response:	boolean	true = successful operation false = error
Notes:	see examples: ---	

List of Examples.

In the router's WEB configuration environment, in the "Other → TITAN Scripts" section, you will find a series of example scripts. We will now provide a brief description of each.

Example 1.- Hello World

Basic example of how to write debug elements for the standard output.

Example 2.- Request AT Command

Example of how to send AT commands to the internal modem of the device (to find out the coverage, identification of the telephone cell used, etc.)

Example 3.- Send SMS when GPIO change

Example of how to send an SMS message when a change in a digital input of the device is detected.

Example 4.- Send MQTT message to broker

Example of how to send an MQTT message to the broker to which the device is connected.

Example 5.- Send SNMP TRAP

Example of how to send an SNMP TRAP.

Example 6.- Send EMAIL when power fails

Example of how to send an email. In this example, the email is sent when a power failure is detected. Example only valid for router models with a built-in battery or supercap.

Example 7.- Read Modbus RTU word register

An example of how to read a Modbus RTU WORD register from TITAN scripts.

Example 7b.- Reading a Modbus TCP word register

An example of how to read a Modbus TCP WORD register from TITAN scripts.

Example 8.- Read Modbus RTU word register and send via MQTT

Example of how to read a Modbus WORD register and send the read data to an MQTT topic.

Example 9.- Read Modbus RTU word register and send TRAP

Example of how to read a Modbus WORD register and send an SNMP TRAP based on the value of the register read.

Example 10.- Read Modbus RTU word register and send SMS

Example of how to read a Modbus WORD register and send an SMS according to the value of the register read.

Example 11.- Read Modbus RTU word register and send EMAIL

Example of how to read a Modbus WORD register and send an EMAIL according to the value of the register read.

Example 12.- Write Modbus RTU word registers

Example of how to write to a Modbus WORD register.

Example 13.- Read Modbus RTU bit register

Example of how to read a bit-type Modbus register.

Example 14.- Write Modbus RTU bit register

Example of how to write to a bit-type Modbus register.

Example 14b.- Reading a Modbus RTU word register and saving data in the LOGGER for scheduled sending

An example of how to read Modbus RTU registers from an external device via an RS485 serial port, create a JSON with a custom data structure and store this JSON in the Titan datalogger to be sent via the configured method (MQTT, HTTP, FTP).

Example 14c.- Convert Modbus word registers into Unsigned Int32, Signed Int32, Float, Int64

Example of how to read modbus RTU registers from an external device via an RS485 serial port as WORD and transform them into 32-bit unsigned integers, 32-bit signed integers, float and 64-bit signed integers.

Example 15.- Read all Digital Inputs

Example of how to read all IOs (digital inputs and/or outputs) of the device. Only valid for devices with digital inputs and/or outputs.

Example 16.- Write Digital Output

Example of how to change the status of a digital output of the device. Only valid for devices with digital outputs.

Example 17.- Ping ip and send SMS if problem with Ping

Example of how to PING to a connected device and send an SMS in case of problems with PING, e.g.: to detect the malfunctioning of a connected device.

Example 18.- Send SMS Alarm when Backup SIM 2 is being used

Example of sending an SMS warning when the secondary SIM becomes operational. Only valid for DUAL SIM devices.

Example 19.- Read and Write data from SERIAL PORT

Example of how to read and write on a serial port. Example only valid for devices with one or more RS232 and/or RS485 serial port(s)

Example 20.- Use of WATCHDOG

Example of how to use the TITAN Scripts watchdog. Restart in case of script malfunction.

Example 21.- Use of LEDs for custom indications

Example of how to use the LEDs in a customised way, managing switching on and off from a script.

Example 22.- TITAN as MODBUS concentrator saving data into internal memory

Example of how to use the device as a Modbus concentrator, saving the read data in an internal memory. For example, it is possible to autonomously read different Modbus RTU sensors and store the read registers in different positions within the memory of the device and, from a remote location, read the data of the registers stored within the device through Modbus TCP.

Example 23.- Exchanging data through AT Commands sent by SMS, MQTT, Telnet, SSH, HTTP ...

Example of how to exchange data via AT commands from scripts sent by SMS, MQTT, Telnet, etc. That is, you can send an AT command via SMS, MQTT, Telnet, etc. to the device, and from the script programmed by you, you will be able to receive such data and act accordingly.

Example 24.- Encrypting the code of Script

Example of how to encrypt the code of a script. For example, if you are going to provide a third party with a device with a pre-programmed script and you do not want to show the source code, you can encrypt the script as shown in this example.

Example 25.- Encrypting the code of Script and avoiding the Piracy of the Script

Example of how to encrypt the code of a script. For example, if you are going to provide a device with a pre-programmed script to a third party and you do not want to show the source code, so that it cannot be copied or hacked, you can encrypt the script as shown in this example.

Example 26.- Saving DATA into the internal TITAN LOGGER.

Example of how to store data in the internal memory of the device. In this way, you will be able to benefit from the automatic mechanisms of the device using its memory to send data to an MQTT broker, HTTP server, etc.

Example 27.- Use of user FILES to read, write and append data.

Example of how to create user files to write, read and append data.

Example 28.- Uploading user FILES by FTP

Example of how to send user files via FTP.

Example 29.- Downloading user FILES by FTP

Example of how to receive user files via FTP.

Example 30.- Read Modbus RTU, save data into FILE, and send it by FTP

Example of how to read Modbus RTU data from a script, save the data to a user file and send the file via FTP.

Example 31.- Download and install new version of SCRIPT from FTP Server

Example of how to remotely update a user script.

Example 32.- Read instant values of IEC-60870-5-102 Meter

Example of how to read the instantaneous values of an energy meter with the IEC- -60870--5--102 protocol from a script.

Example 33.- Read GPS data from TITAN

Example of how to obtain the GPS position from a script. Only valid for devices with GPS.

Example 34.- Read digital METERS

Example of how to read the value of the pulse meter. Only valid for devices with digital inputs.

Example 35.- Subscribe to MQTT topic. Read and Send byte array data

Example of how to subscribe to an MQTT topic to receive a data array.

Example 36.- IO Line Passing by MQTT. Send digital input status to remote digital output.

Example designed to use 2 TITAN-based devices with digital inputs / outputs. When there is a change of status in a digital INPUT of an device, the status of a digital OUTPUT will be changed in the remote device. That is, it shows how to replicate a digital input to a remote digital output.

Example 37.- Check and download a remote router Configuration via FTPS.

This is an example of how to make a TITAN-based device download a configuration backup file via FTPS. The router downloads the file, renames it on the remote server (to prevent it from downloading again) and installs the downloaded configuration.

Example 38.- Download a remote file containing AT commands and execute them via FTPS

This is an example of how to make a TITAN-based device download a file containing AT commands via FTPS. Once the file has been downloaded, the TITAN-based device will execute the AT commands read from the file.

Example 39.- Send data via HTTP GET and HTTP POST with custom headers

This is an example of how to make HTTP/HTTPS requests using GET and POST methods, which also enables the use of custom headers.

Example 40.- Read all received SMS from SCRIPT

Example of how to read all SMSs received by the TITAN-based device from a script. From the script, once the SMS has been received, you can obtain the telephone number of the SMS sender and the corresponding text message.

Example 41.- Send TELEGRAM message when a GPIO changes.

Example of how to send a text message to a group chat on the Telegram messaging application when a digital input of the TITAN-based device changes. See the application notes for more information.

Example 42.- Send TELEGRAM message when a MODBUS register changes.

Example of how to send a text message to a chat group on the Telegram messaging application when changing the value of a Modbus register read by the TITAN-based device via Modbus RTU from a device connected via the RS485 port. See the application notes for more information.

Example 43.- Read CellID and get approximate Latitude and Longitude from third party.

Example of how to obtain the latitude and longitude from the information obtained from the telephone cell used by the modem. A third-party API is also used.

Example 44.- Read the cheapest energy hour from REE website by HTTP REST API.

Example of how to download, from REE (Red Eléctrica de España) via HTTP GET (REST API), the hourly rates of the regulated electricity market in Spain. Once the hourly energy price tariffs have been downloaded, the device will look for the hour with the lowest energy price. The digital output of the device will only be enabled during that hour (e.g.: to activate an electric hot water boiler) to reduce the electricity bill as much as possible.

Example 45.- Reading and sending data through a TCP Client SOCKET.

An example of how to connect a TCP Client socket to a server and how to send and receive data through it.

Example 46.- Reading and sending data through a TSSL/TLS CP Client SOCKET.

An example of how to connect a SSL/TLS-secured TCP Client socket to a server and how to send and receive data through it.

Example 47.- Checking for the presence of an IEC-60870-5-102 meter and sending SMS alarm notifications if undetected

An example of how to periodically check for the presence of an IEC-60870-5-102 meter. If the meter isn't present, you will receive an SMS alarm notification to inform you of the situation.

Example 48.- Reading instant values of several IEC-60870-5-102 meters and storing data in the LOGGER

An example of how to read the instant values of various energy meters with the IEC-60870-5-102 protocol from a script. Data will be stored in the internal LOGGER so that it can then be sent to a WEB platform via HTTP, MQTT or FTP.

Example 49.- Configuring the IEC-60870-5-102-to-MODBUS TCP/RTU Gateway conversion

IEC-60870-5-102 to Modbus converter. An example of how to read the instant values of a meter with the IEC-60870-5-102 protocol and store them in the user memory area so that they can be read via Modbus TCP or Modbus RTU by an external device (e.g. a PLC).

Example 50.- Communicating with an ANDROID app using MQTT

An example of how to interact from a script with an Android app designed by the user. You must see application note 61.

Example 51.- Forwarding all received SMS messages via the RS232 serial port

An example of how to receive an SMS message from a script and forward the contents of such message directly via the RS232 serial port.

Example 52.- Reading INTEGRATED values of IEC-60870-5-102 meters and storing data in the LOGGER

An example of how to read the load curves (integrated values) of an IEC-60870-5-102 meter every day at 2 am and store the data in the LOGGER memory so that the data can be configured automatically (MQTT, HTTP, FTP).

Example 53.- Configuring the DLMS/ COSEM-to-MODBUS TCP/RTU Gateway conversion

DLMS/COSEM to Modbus converter. An example of how to read the instant values of a meter with the DLMS/ COSEM protocol and store them in the user memory area so that they can be read via Modbus TCP or Modbus RTU by an external device (e.g. a PLC).

Example 54.- Reading instant values of a DLMS COSEM meter and storing data in the LOGGER

Example of how to read instant energy values with the DLMS-/ COSEM protocol from a script. Data will be stored in the internal LOGGER so that it can then be sent to a WEB platform via HTTP, MQTT or FTP.

JSON Transformer Script Function

The JSON Transformer script function allows you to change the data format before sending it to a WEB platform via HTTP, MQTT or FTP. In this way, you can adjust the format of the data sent by the TITAN-based device to the one required by the platform. See Application Note ANV6_57-Router-TITAN-Json_Transformer_Script for full details on this feature.

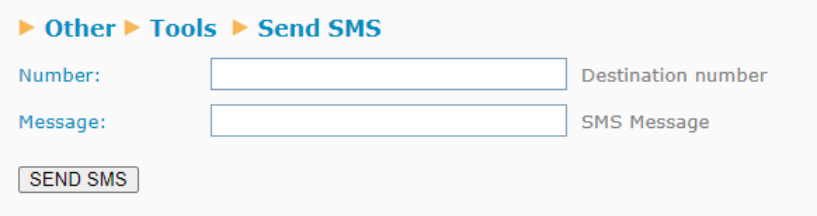
MQTT Topic Function Script

This feature lets you run a script when the TITAN-based device receives data on specific MQTT topics. In this way, it can receive and process data received from an MQTT web platform regardless of the data format. Refer to the application note ANV6_58- Router-TITAN-Mqtt_Topic_Function_Script for full information on this feature.

4.7.19- Other connectivity tools

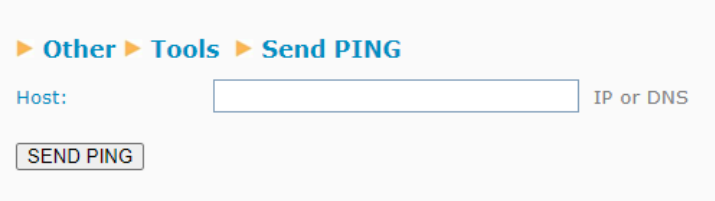
Sometimes, it is useful to know the number of the SIM card inserted in the TITAN-based device or to check connectivity against a given IP or DNS from the device's point of view, or to check connectivity with the TCP port of an external server, or even just to know if the email configuration section has been set up correctly. This section provides a set of basic tools to meet all these needs.

- “Send SMS" feature.



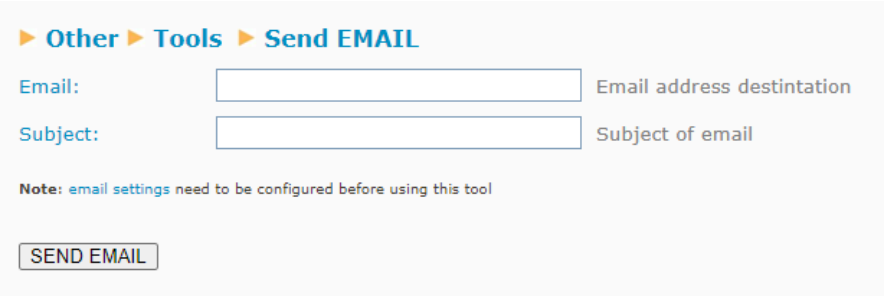
As its name suggests, this feature makes it very easy to send an SMS message from a TITAN-based device to a mobile phone. It allows you to quickly obtain the phone number corresponding to the SIM card.

- “Send PING" feature.



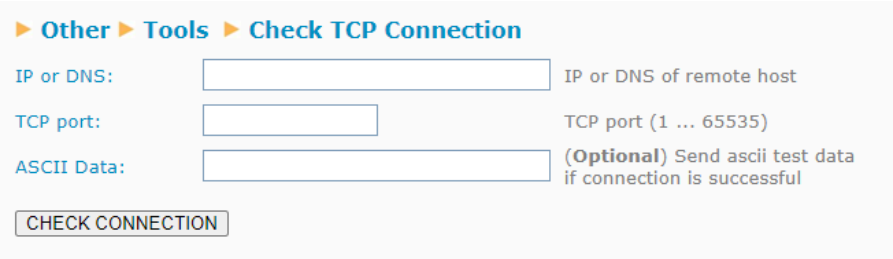
You can PING from the TITAN-based device's point of view to a given IP or DNS using this feature, allowing you to check the connectivity between both entities.

- “Send EMAIL" feature.



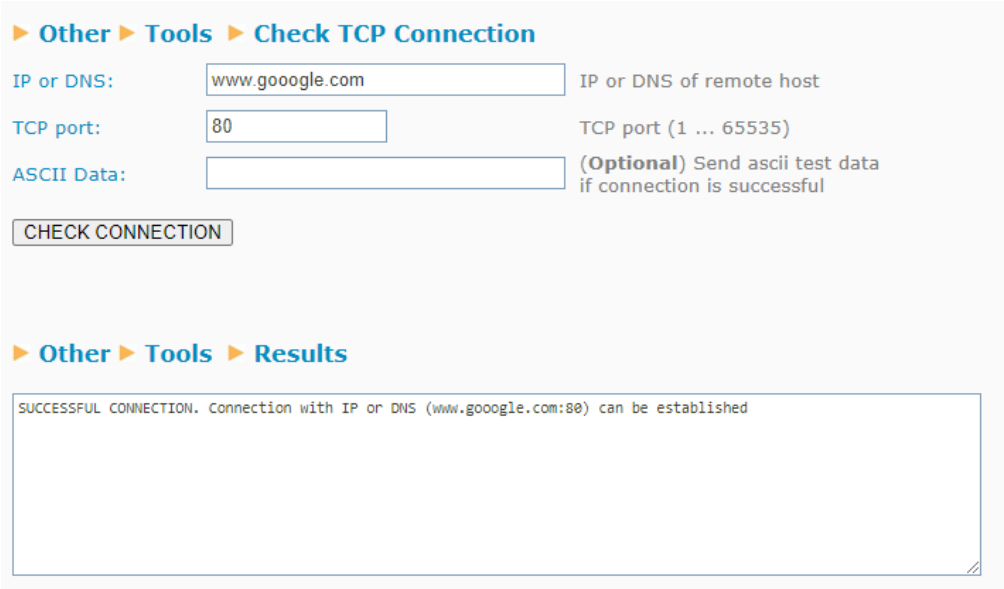
This feature allows you to check the correct configuration for sending emails from TITAN-based devices. To do this, the "Other→Email" section must first be configured.

- "Check TCP Connection" feature.



This feature allows you to check if a TCP port is available for connection from the point of view of the TITAN-based device. To check this connectivity, the device creates a TCP Client socket, which attempts to connect to the specified IP/DNS and TCP port. If the connection is made successfully, the result will be shown as successful. You also have the option to add an additional ASCII data field, which the TITAN-based device will send via this socket once the connection has been made.

There is a "Results" section at the bottom of the screen, where the results of the above tools are displayed. The following example shows the result of an attempt to connect to the address www.google.com and the TCP 80 port.


Other ▶ Tools ▶ Check TCP Connection
IP or DNS:
www.gooogle.com
IP or DNS of remote host
TCP port:
80
TCP port (1 ... 65535)
ASCII Data:
(Optional) Send ascii test data if connection is successful
CHECK CONNECTION
Other ▶ Tools ▶ Results
SUCCESSFUL CONNECTION. Connection with IP or DNS (www.gooogle.com:80) can be established

4.7.20 - Other >> Digital I/O

Many models of modems and TITAN-based devices have digital inputs and/or outputs, and you can configure them in this section.

- Enabled: enables the TITAN-based device's Input/Output service. You must enable the service if you intend to use the inputs and outputs.
- Period: allows you to set a time period (in minutes) within which to read the digital inputs and outputs and pulse meters, and to store the readings in the internal datalogger of the TITAN-based device to be sent later to a platform via HTTP/S or MQTT/S.

Once the option has been enabled and the device has been restarted, you can configure the behaviour of the digital inputs and digital outputs by pressing the "CONFIG" button corresponding to each of them. Remember that digital inputs and outputs can also be managed from the TITAN Script.

Configuration of the digital inputs.

The configuration display for the digital inputs looks like this.

- ***Digital input mode***: indicates the behaviour of the digital input. You can choose between "normal", if you do not wish to use the input for alarms, and "Alarm (pin 0 → 1)" or "Alarm (pin 1 → 0)", if you wish to configure an alarm to be triggered when the logic value of the digital input changes from 0 to 1 or vice versa.
- ***SMS enabled***: must be activated if the TITAN-based device is supposed to send an SMS message when an alarm is activated or deactivated.
- ***Phone numbers***: please enter the telephone numbers to send SMS messages to here. If you need to send them to more than one telephone number, you can indicate several telephone numbers separated by ";".

- ***Test SMS Alarm On***: SMS text message to be sent when the alarm is activated.
- ***Test SMS Alarm Off***: SMS text message to be sent when the alarm is deactivated.
- ***Logger enabled***: must be enabled if the TITAN-based device is supposed to store the change of the digital input in the Logger, so that it can be sent to a platform via HTTP/HTTPs, MQTT/MQTTS or FTP.

If the LOGGER is enabled, this is the JSON object that will be sent to the server.

{"IMEI":"867962046823806","TYPE":"GPIO","TS":"2021-11-03T08:16:00Z","ID":0,"VALUE":1,"DIR":"INPUT","P":"ID0001"}

Where:

IMEI: the modem's IMEI

TYPE: type of frame. In this case, "GPIO"

TS: Timestamp of when the event occurred

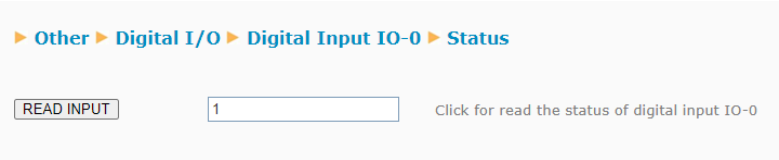
ID: IO identifier (0, 1, etc.)

VALUE: logic value of the digital input (0, 1)

DIR: indicates whether the GPIO is of type INPUT or OUTPUT

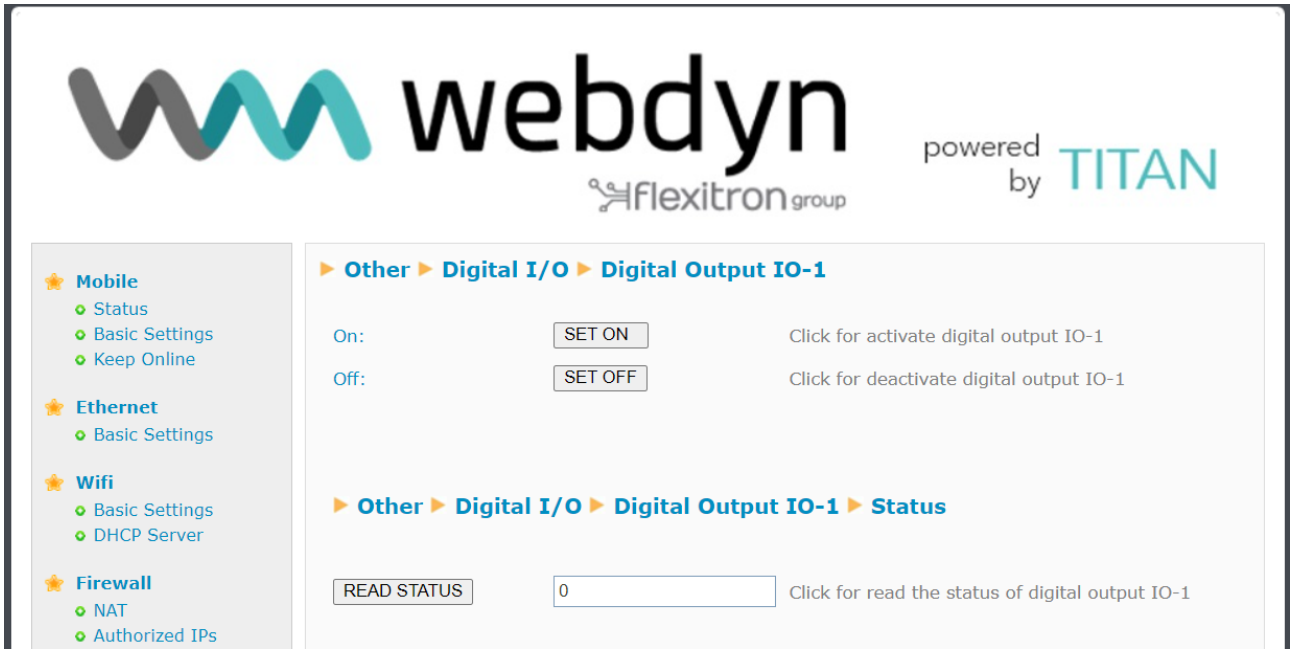
P: ID field configured in LOGGER section

It is also possible to read the value of the digital input in real time by pressing the "READ INPUT" button.



Configuration of the digital outputs.

The configuration display for the digital outputs looks like this



From this screen, it is possible to change the status of a digital output and read the current status of the output. Remember that it is also possible to change the status of a digital output from TITAN Scripts.

4.7.21- Other → Custom Skin

TITAN-based devices allow you to configure the header of the web configuration environment with your own logos, as well as the title and footer of the same environment. This gives the device the appearance of a customised product or white label to a certain extent.

- Set custom image: select an image of 922 x 172 pixels (gif type) for the header area.
- Set custom labels: enter the text that you want to appear in the title and footer of the web configuration environment.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes.
- Please note that if you configure the device with your customised header and texts to be seen by third parties, it will not make sense for you to use the "Other > User Permissions" section. Use this section to prevent the user ("user") from accessing the "Other > Custom Skin" section.

4.7.22- Other → Custom Led

TITAN-based devices have several configurable LEDs. In this section, you can configure their behaviour to best suit their needs.

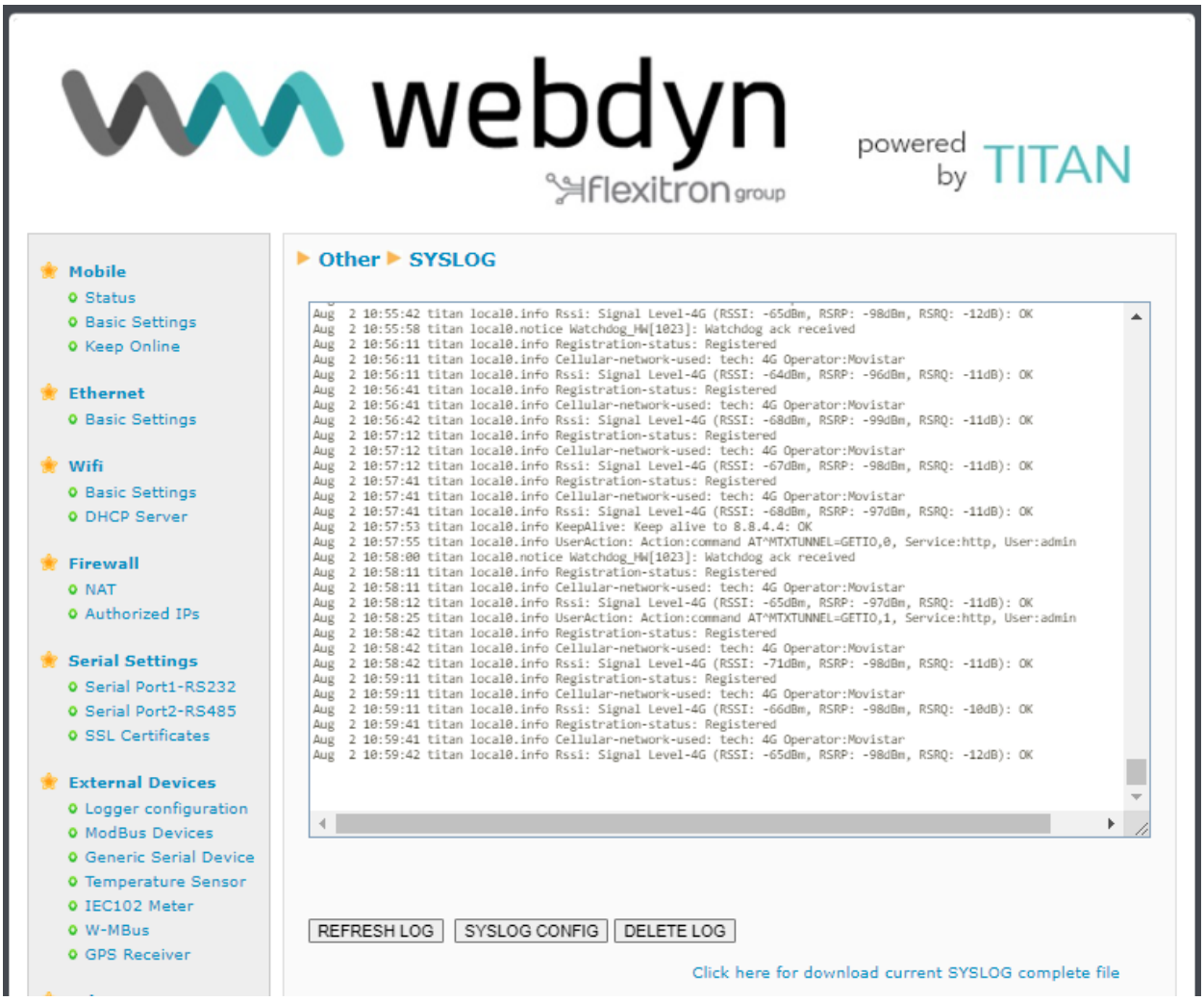
- ***Not used:*** the LED will always be off and unused.
- ***IP:*** the LED will light up if the device has an IP address (WAN 4G/3G/2G).
- ***RSSI:*** the LED will flash when coverage is low and will remain on without flashing when coverage is good. Off indicates critical or no coverage.
- ***SIM Error:*** the LED will light up when a SIM card failure is detected.
- ***TITAN Scripts***: the LED is going to be used in the TITAN Scripts section of the device.

Additional Notes.

- Once the configuration process is finished, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.7.22- Other → SYSLOG

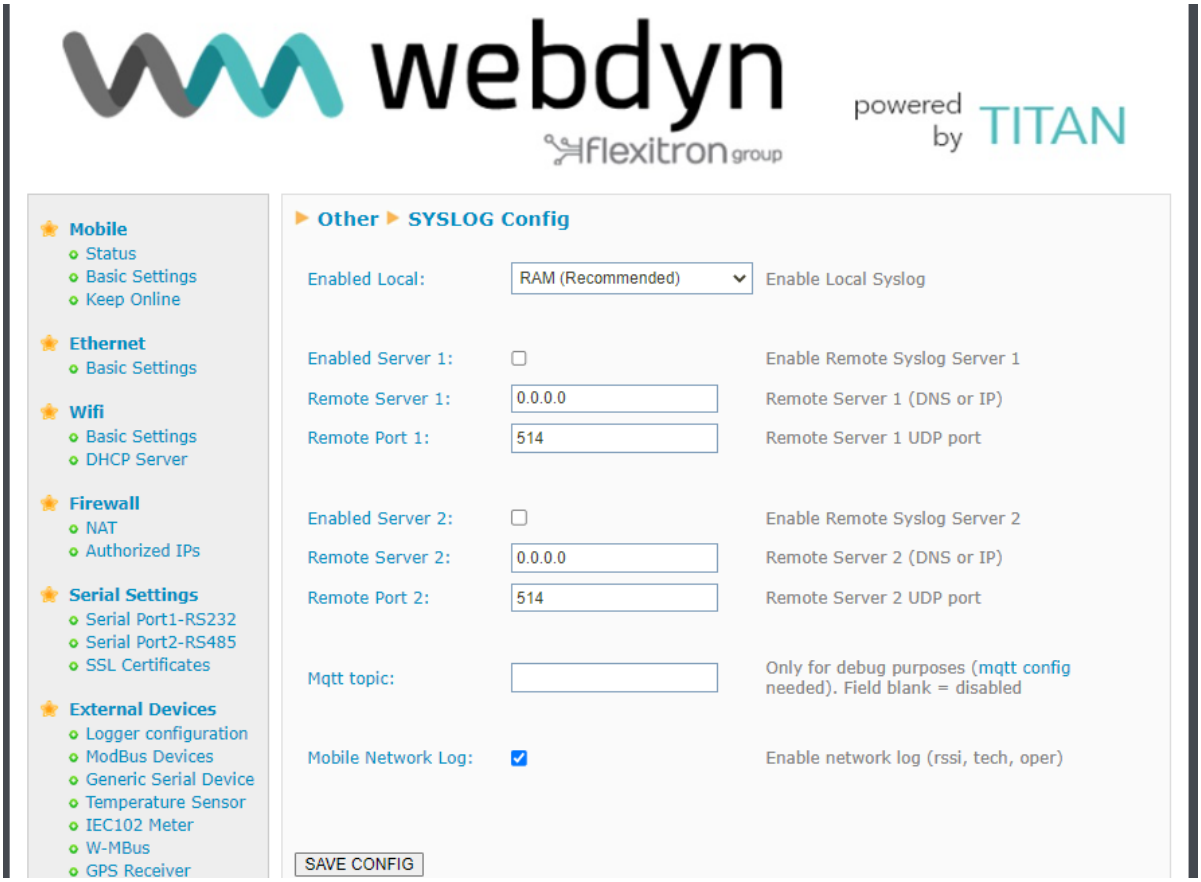
In this section, you can access an internal log of the TITAN-based device, where you will be able to see the status of the device itself (coverage, network registration, operator, result of sending data to servers...), the actions carried out by users, etc.



Click on the **"SYSLOG CONFIG"** button to access the SYSLOG configuration section. You can set the following configurations in this configuration section.

- ***Enabled Local:*** "Disabled": will not use the SYSLOG. "RAM" (recommended option) will store the SYSLOG in the RAM memory. "FLASH" (only available on some TITAN-based devices) will save the SYSLOG in flash. Only recommended for debugging and not for permanent use, since its constant use can excessively wear out the FLASH memory.
- ***Enabled Server 1:*** checking the box enables the SYSLOG to be sent to a remote server via UDP.
- ***Remote Server 1:*** IP or DNS of remote server 1 to receive the SYSLOG

- ***Remote Port 1:*** Remote server 1 USD port
- ***Enabled Server 2:*** checking the box enables the SYSLOG to be sent to a remote server via UDP.
- ***Remote Server 2:*** IP or DNS of remote server 2 to receive the SYSLOG
- ***Remote Port 2:*** Remote server 2 USD port
- ***MQTT Topic:*** If an MQTT topic is specified, the real-time syslog will be sent to the specified MQTT topic. The Other → MQTT section must be configured.
- ***Network Log:*** checking the box adds the network status to the SYSLOG on a regular basis.



Additional Notes.

- Once the configuration process is finished, click on the "SAVE CONFIG" button to save the changes. Remember that you must restart the device for the new changes to take effect.

4.7.23- Other → Backup / Factory.

You can make a full backup of the device configuration in this menu. You can save the configuration to a file and restore it back to the device when needed. You can also re-set the device to factory settings.

- ***"Factory Settings" button:*** press this button if you wish to restore the device to the default factory configuration.
- ***"Download settings"*** button: click on the button to download the device configuration as a file named "config.mtx".
- ***"Select file" button:*** to restore a saved configuration, after indicating the configuration file to be used, press the "Upload" button to upload the file.

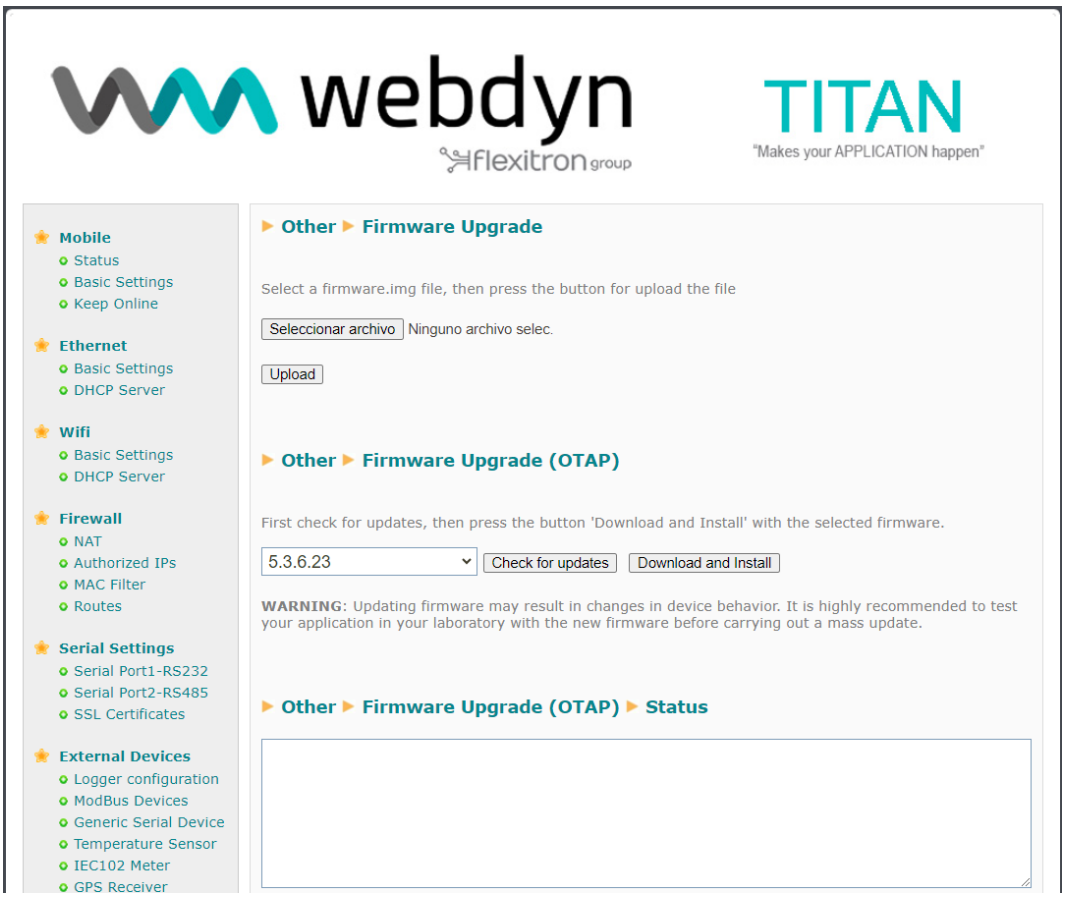
4.7.24- Other → Reboot.

In this section, you can restart the device. For example, to apply the changes after a configuration change. To do so, simply press the 'Reboot" button

Depending on the device, the "Power down" option is also available. With this option, you can switch off a device model with a built-in battery, such as the MTX-Router-Q model. Before pressing the "Power down" button, you must disconnect the power supply (220Vac) to the device, and then press the button. After a few seconds, all LEDs will remain off.

4.7.25- Other → Firmware Upgrade

From this section, it is possible to upgrade the Titan router's firmware either locally or remotely.



Firmware Upgrade

If you have the firmware file to update the device, click on the "select file" button, select the file, and then click on the "Upload" button. This process can take up to two minutes and the Titan router may reboot once or twice at the end of the process.

Firmware Upgrade (OTAP)

Another option to update the device's firmware is through the OTAP process. If you intend to use this option, it is necessary that the Titan router has access to the Internet, as the firmware will be downloaded from Webdyn's servers. To carry out the OTAP process, click on the "Check for updates" button. After a few seconds, if there are firmware updates for your router, they will appear in the drop-down menu. Select the firmware version you wish to install and click on the "Download and Install" button to start the process.

Remember that any firmware change process on any device carries risks. It is highly recommended that you verify that the scenario you have configured on the Titan router works correctly in the lab

with the firmware version you intend to install before proceeding with mass firmware upgrades of remotely located devices.

4.8.1- VPN → OpenVPN Server

TITAN-based routers can act as an OpenVPN Server. By establishing a VPN, all devices connected to the Ethernet bus of the device, or even IP-RS232/485 gateways, will be securely accessible from the other end of the VPN network. A VPN infrastructure is very useful to circumvent most problems with proxies, firewalls, etc., especially using SIM cards from an operator that issues private IP addresses (10.x.x.x.x)

- ***Enabled:*** check this box if you wish to enable the OpenVPN service in Server mode
- ***Mode:*** allows you to set the VPN connection mode. The setting "Always On" will keep the VPN enabled at all times. The "Under Request" setting will enable the VPN for as long as you decide. For example, you may be interested in enabling the VPN only at very specific times for maintenance work. When you wish to enable the VPN, you only need to send an AT command (see chapter 5 on AT commands) via SMS, or by Telnet, Webserver or SNMP, or even by Modbus TCP. After the configured time has elapsed, the VPN will be disabled.
- ***Protocol*** You can set the establishment protocol. This can use both UDP and TCP protocols.
- ***Port:*** the standard port is 1194, but you can specify the port for the VPN that you feel is most appropriate.

- ***LZO compression:*** check the box if you wish to enable this option.
- **Auth**: allows you to choose the authentication method.
- **Cipher**: allows you to choose the encryption algorithm.
- ***Server Subnet****:* specifies the subnet to be created after VPN creation. For example, if you specify the subnet 10.8.0.0, the TITAN-based device will adopt the address 10.8.0.1 when the VPN is enabled.
- ***Server Mask***: specifies the netmask to be applied to the VPN
- ***Allow Lan Access:*** check the box if you wish to remotely access devices connected to the Ethernet port of your device. Do not enable this feature if you only wish to access the TITAN-based device and the IP-RS232/485 gateways it manages.
- **Files required for OpenVPN**

  In order to use TITAN-based devices as an OpenVPN server, you will need to create and upload the files ca.crt (authority certificate), server.crt (server certificate), **server.key** (server private key), and **dh1024.pem** (Diffie Hellman parameters).

  At the bottom of the screen, you will be able to download demo files that should only be used for testing purposes. Remember that in a real application, you should generate your own certificates.

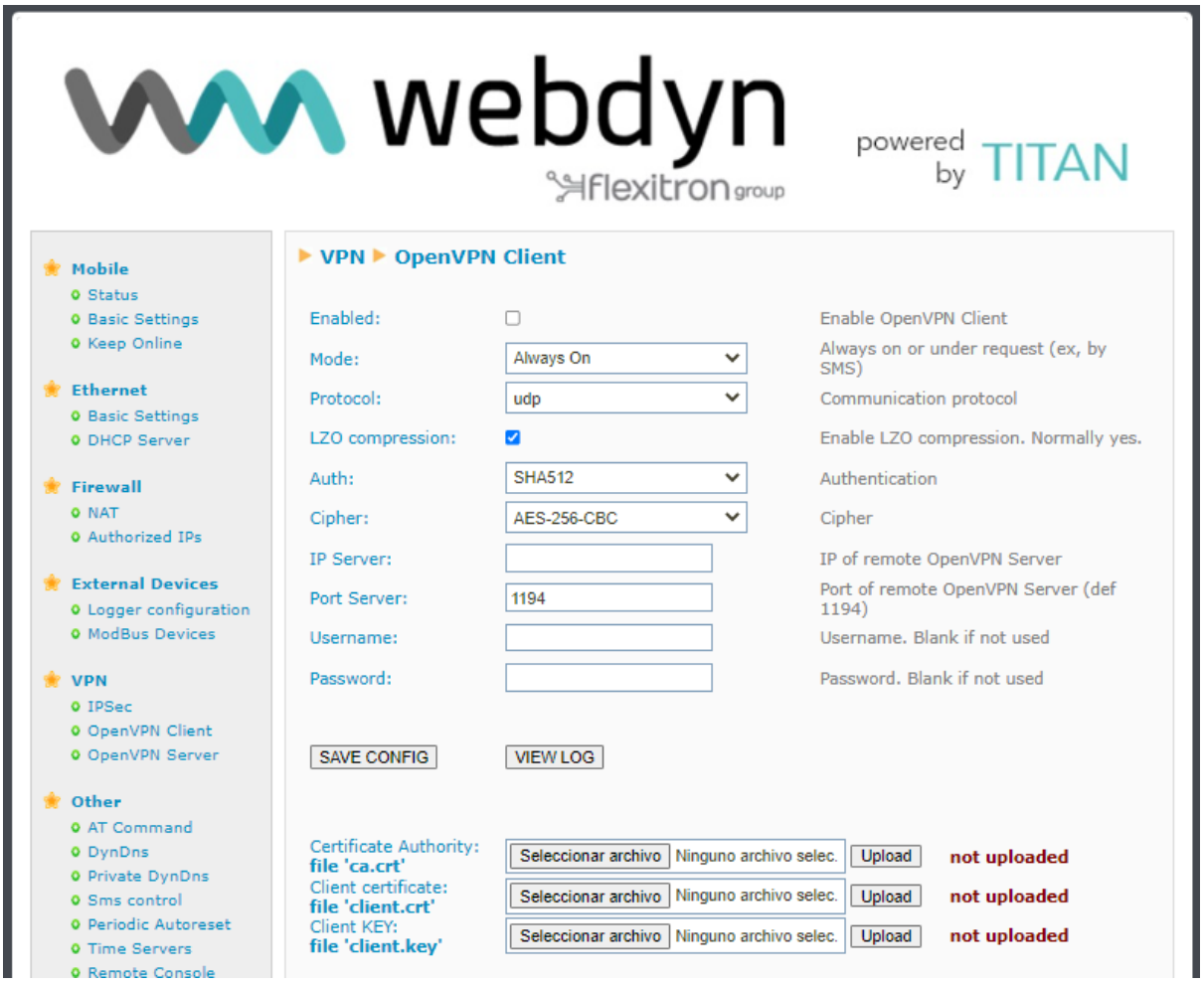
Additional Notes.

- Once the configuration process is complete, click on the "SAVE CONFIG" button to save the changes. Remember that you must restart the device for the new changes to take effect.

- Application notes (PDF documents external to this document) including examples of OpenVPN functionality with TITAN-based devices are available.

4.8.2- VPN → OpenVPN Client

TITAN-based routers can act as an OpenVPN Server. By establishing a VPN, all devices connected to the Ethernet bus of the device, or even IP-RS232/485 gateways, will be securely accessible from the other end of the VPN network. A VPN infrastructure is very useful to circumvent most of the problems with proxies, firewalls, etc., especially using SIM cards from an operator that issues private IP addresses (10.x.x.x.x)



- ***Enabled:*** check this box if you wish to enable the OpenVPN service in Client mode
- ***Mode:*** allows you to set the VPN connection mode. The setting "Always On" will keep the VPN enabled at all times. The "Under Request" setting will enable the VPN for as long as you decide. For example, you may be interested in enabling the VPN only at very specific times for maintenance work. When you wish to enable the VPN, you only need to send an AT command (see chapter 5 on AT commands) via SMS, or by Telnet, Webserver or SNMP, or even by Modbus TCP. After the configured time has elapsed, the VPN will be disabled.
- *Protocol:* you can set the establishment protocol. This can use both UDP and TCP protocols.

- **Port**: the standard port is 1194, but you can specify the port for the VPN that you feel is most appropriate.
- **LZO compression**: check the box if you wish to enable this option.
- ***Auth:*** allows you to choose the authentication method.
- **Cipher**: allows you to choose the encryption algorithm.
- ***IP Server:*** enters the Public IP address of the remote server that will act as the OpenVPN server.
- *Files required for OpenVPN*

  In order to use TITAN-based devices as an OpenVPN client, you will need to create and upload the files ca.crt (authority certificate), **client.crt** (client certificate) and **client.key** (client private key).

  At the bottom of the screen, you will be able to download demo files that should only be used for testing purposes. Remember that in a real application, you should generate your own certificates.

Additional Notes.

- Once the configuration process is complete, click on the "SAVE CONFIG" button to save the changes. Remember that you must restart the device for the new changes to take effect.

- Application notes (PDF documents external to this document) including examples of OpenVPN functionality with TITAN-based devices are available.

4.8.3- VPN → IPSec

TITAN-based routers may come with IPSEC. By establishing a VPN, all devices connected to the Ethernet bus of the device, or even IP-RS232/485 gateways, will be securely accessible from the other end of the VPN network. A VPN infrastructure is very useful to circumvent most of the problems with proxies, firewalls, etc., especially using SIM cards from an operator that issues private IP addresses (10.x.x.x.x)

- ***Enabled:*** check this box if you wish to enable the IPSec service
- *ipsec.conf:* enter your IPSec VPN configuration file here. You can view and modify the examples most suited to your needs at the bottom of the screen.
- ipsec.secrets: enter the contents of the ipsec.secrets file of your IPSec VPN here. You can view and modify the examples most suited to your needs at the bottom of the screen.

TITAN-based device acting as an IPSec client:

- xca1-cert.pem: file with the root CA certificate of your IPSEC server
- xclient1-cert.pem: file with the client certificate

- xclient1-key.pem: file with the KEY

TITAN-based device acting as an IPSec server:

- *ca-cert.pem:* root CA file used by the TITAN-based device
- ca-key.pem: file with the CA KEY of the TITAN-based device
- server-cert.pem: certificate file of the TITAN-based device
- server-key.pem: file with the KEY of the TITAN-based device
- client1-cert.pem: file with the client 1 certificate, authorised to connect
- client2-cert.pem: file with the client 1 certificate, authorised to connect
- client3-cert.pem: file with the client 1 certificate, authorised to connect

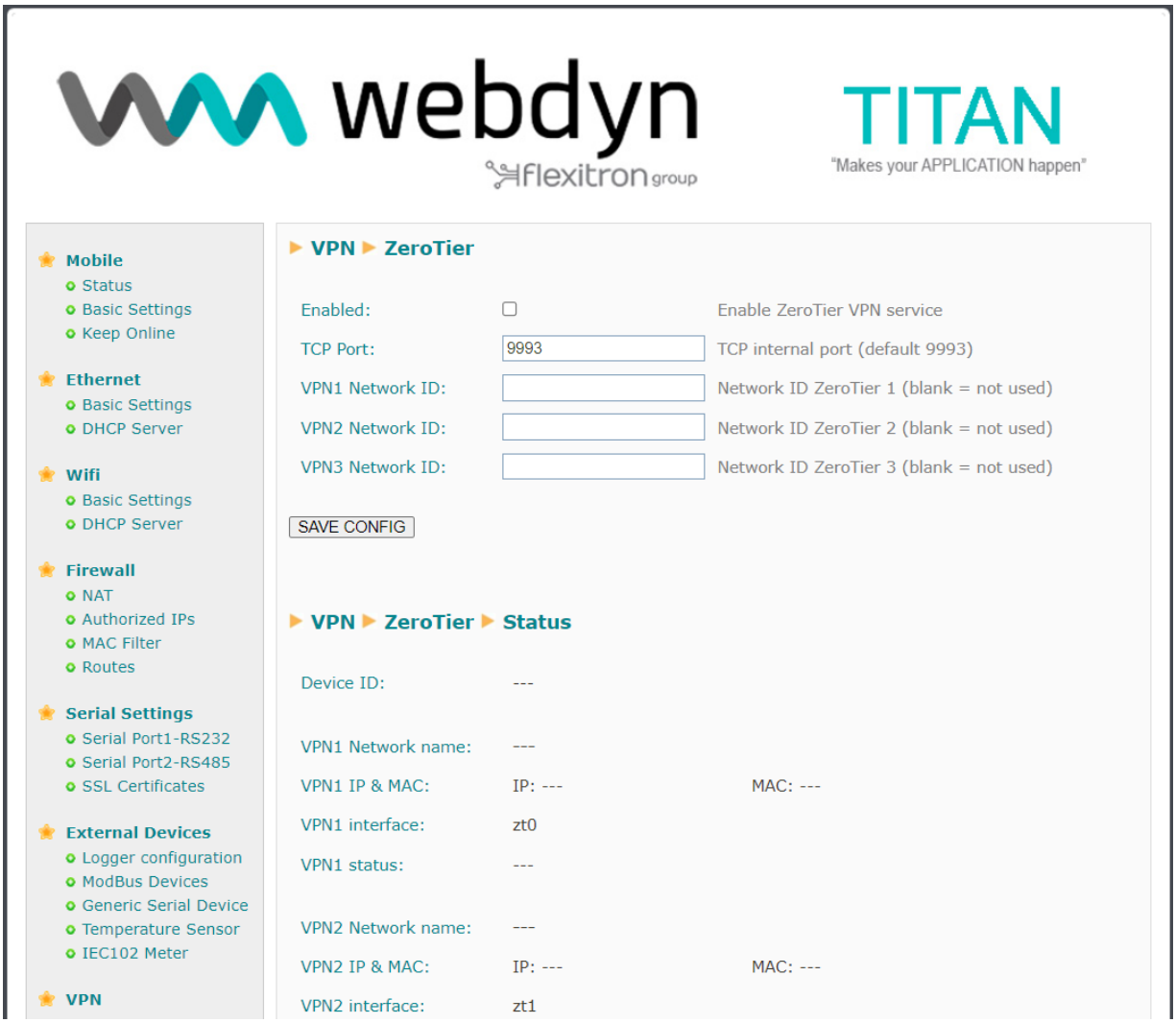
Additional Notes.

- Once the configuration process is complete, click on the “SAVE CONFIG” button to save the changes. Remember that you must restart the device for the new changes to take effect.

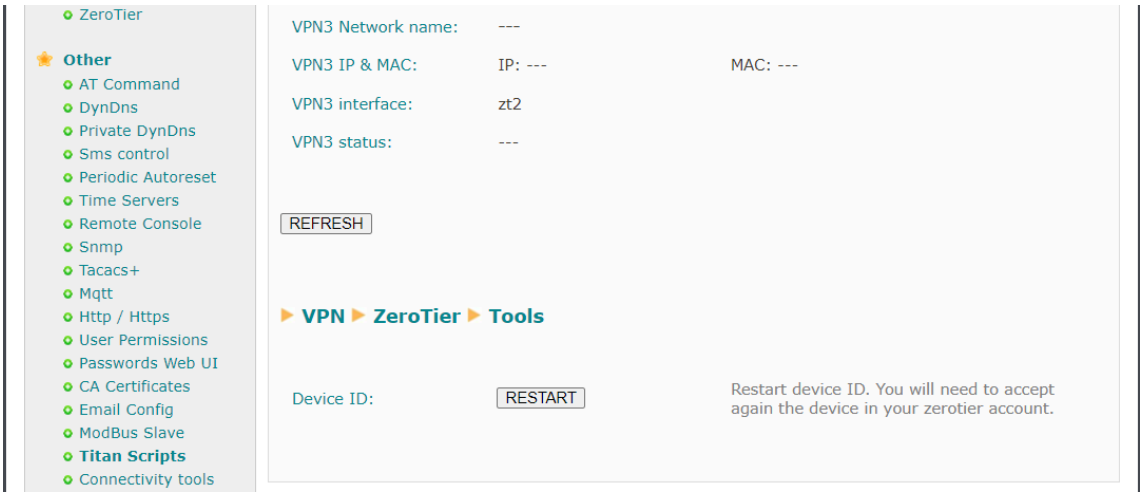
- Application notes (PDF documents external to this document) including examples of IPSec functionality with TITAN-based devices are available.

4.8.4 VPN → Zerotier

The FW version 5.3.6.25 and more recent versions of Titan routers come with VPN support for the ZeroTier service. ZeroTier allows you to deploy a VPN very easily and quickly, enabling access to the Titan router and the devices connected to it from any location. See application note AN71 for examples of use and more detailed information.



- Enabled: Check this box if you wish to enable the IPSec service
- TCP Port: TCP port for internal use of the Titan router. Keep the default TCP port 9993 if you do not need it for another service.
- *VPN1 Network* 1: ZeroTier identifier of VPN network 1
- *VPN1 Network* 2: ZeroTier identifier of VPN network 2
- *VPN1 Network* 3: ZeroTier identifier of VPN network 3



- *Restart*: Button to change the Device ID of the ZeroTier network device. If the Device ID has been changed, you will need to go to the ZeroTier control panel (my.zerotier.com) and re-authorise the device on the network.

5.- AT commands

TITAN-based devices allow AT commands to be sent directly to the internal modem through multiple interfaces:

- 1.- Via a serial port.
- 2.- Via a 4G/3G/2G-Serial Gateway (via IP WAN, Ethernet or Wifi) through embedded AT commands.
- 3.- By SMS
- 4.- Via Telnet (Remote Console, via 4G/3G/2G, Ethernet or Wifi)
- 5.- SSH (Remote Console, via 4G/3G/2G, Ethernet or Wifi)
- 5.- Via Webserver (via 4G/3G/2G, Ethernet or Wifi)
- 7.- Via Modbus RTU (via the RS232 or RS485 serial port)
- 8.- Via Modbus TCP (via 4G/3G/2G, Ethernet or Wifi)
- 9.- Via SNMP (via 4G/3G/2G, Ethernet or Wifi)
- 10.- Via URL. Explained at the end of this point.

You can therefore send AT commands to the device at your own risk.

- **AT^MTXTUNNEL=REBOOT**

  Action: re-set the TITAN-based device.

  Result:
  OK: Command executed correctly.
  ERROR: Command executed, but with an error.

  Example: AT^MTXTUNNEL=REBOOT
  OK

- **AT^MTXTUNNEL=VERSION**

  Action: returns the firmware version of the TITAN-based device.

  Result:
  OK: Command executed correctly.
  ERROR: Command executed, but with an error.

  Example: AT^MTXTUNNEL=VERSION 5.2.6.17 OK

- **AT^MTXTUNNEL=GETIP**

Action: returns the WAN IP address (2G/3G/4G)

Result:

OK: Command executed correctly.

ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=GETIP 88.28.221.14 OK

- **AT^MTXTUNNEL=GETIMEI**

Action: returns the IMEI of the internal modem

Result:

OK: Command executed correctly.

ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=GETIMEI 869101054255506 OK

- **AT^MTXTUNNEL=GETCELLID**

Action: returns the telephone cell being used by the modem.

Result:

OK: Command executed correctly.

ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=GETCELLID

[4;214;07;219B;15F2D2A]

OK

Where: 4 = technology used (2=2g,3=3)

214 = MCC (Mobile Country Code)

07 = MNC (Mobile Network Code)

07 = LAC (Local Area Code)

15F2D2A = CID (Cell ID)

AT commands related to messaging:

- **AT^MTXTUNNEL=TRAP,<OID>;<myMessage>;<mySeverity>**

  Action: allows an SNMP TRAP to be sent, with a given OID, the corresponding message and the severity.

  Parameters:

  <OID>: OID of the trap to be sent by SNMP <myMessage>: text message to be sent <mySeverity>: 0 ... 7 trap severity

  Result:

  OK: Command executed correctly. Message sent.
  ERROR: Command executed, but with an error.

  Example: AT^MTXTUNNEL=TRAP,.1.3.6.1.4.1.45711.1.1.11.1.1;myMessage;5
  OK

- **AT^MTXTUNNEL=SMS,<telephoneNumber>,<message>**

  Action: allows you to send an SMS message to a specific telephone number.

  Parameters:

  <telephoneNumber>: phone number to which the SMS message <message> will be sent: text message to be sent

  Result:

  OK: Command executed correctly. SMS message sent to the outgoing queue.
  ERROR: Command executed, but with an error.

  Example: AT^MTXTUNNEL=SMS,+34677123456,alarma de robo
  OK

- **AT^MTXTUNNEL=EMAIL,<destinationAddress>,<subject>**

Action: allows you to send an email (subject only) to a specific email address. To send the email, you must have previously configured the “Other > Email Configuration” section.

Parameters:

<destinationAddress>: destination email address <subject>: message to be sent by email

Result:

OK: Command executed correctly. Email sent.
ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=EMAIL,jgallego@matrix.es, Temperature Alarm
OK

- **AT^MTXTUNNEL=TELEGRAM,SEND,<tokenID>,<chatID>,<message>**

Action: allows you to send a Telegram text message to a specific Telegram group chat. You must create a bot with its tokenID and obtain the chatID of the chat to which the message is to be sent. Read the application notes for more information.

Parameters:

<tokenID>: token identifier <chatID>: chat identifier <message>: message to be sent by telegram

Result:

OK: Command executed correctly. Message sent.
ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=TELEGRAM,SEND,5416419553:ZZHof6enVx-y7uFQrL15cpf3QCHqNHp7Dtg,-730297962,My message
OK

I/O-related AT commands:

- **AT^MTXTUNNEL=GETIO,<idIO>**

  Action: returns the value of a digital input or output, where idIO is the identifier of the IO (0,1,2…)

  Parameters:

  <idIO>: 0, ... X number of I/Os

  Result:

  OK: Command executed successfully, returning the status of the digital input or output.
  ERROR: Command executed, but with an error.

  Example: Status reading for the IO digital input 0

  AT^MTXTUNNEL=GETIO,0
  1
  OK

- **AT^MTXTUNNEL=SETIO,<idIO>,<value>**

  Action: allows you to change the status of a digital output, where idIO is the IO identifier (0,1,2…) and measure the logic value of the output (0, 1)

  Parameters:

  <idIO>: 0, ... X number of I/Os <value>: 0.1 value to be set in the digital output

  Result:

  OK: Command executed correctly. Digital output updated.
  ERROR: Command executed, but with an error.

  Example: AT^MTXTUNNEL=SETIO,3,1
  OK

- **AT^MTXTUNNEL=GETCOUNTER,<idIO>**

Action: returns the value of the pulse reading linked to a digital input, where idIO is the identifier of the IO (0,1,2, etc.)

Parameters:

<idIO>: 0, ... X number of I/Os

Result:

OK: Command executed correctly, returning the value of the pulse meter linked to the digital input

ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=GETCOUNTER,0
1022
OK

- **AT^MTXTUNNEL=SETCOUNTER,<idIO>,<value>**

Action: returns the value of the pulse reading linked to a digital input, where idIO is the identifier of the IO (0,1,2, etc.) and measures the meter reading

Parameters:

<idIO>: 0, ... X number of I/Os <value>: 0.1 value to be set in the meter linked to the digital input.

Result:

OK: Command executed successfully
ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=SETCOUNTER,3,1000
OK

AT commands related to OpenVPN:

- **AT^MTXTUNNEL=OVPNS,minutes**

Action: if you have configured the VPN > OpenVPN Server section as an on-demand OpenVPN, you can enable the OpenVPN in server mode for the specified minutes using this AT command.

Parameters:

<minutes>: 1 … 525,600 minutes

Result:

OK: Command executed successfully
ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=OVPNS,5

(This example would enable the VPN for 5 minutes)

- **AT^MTXTUNNEL=OVPNC,<minutes>**

Action: if you have configured the VPN > OpenVPN Client section as an on-demand OpenVPN, you can enable the OpenVPN in client mode for the specified minutes using this AT command.

Parameters:

<minutes>: 1 … 525,600 minutes

Result:

OK: Command executed successfully
ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=OVPNC,5

(This example would enable the VPN for 5 minutes)

AT commands related to connectivity testing:

- **AT^MTXTUNNEL=COMMAND,<timeout>,<command>**

Action: allows certain special commands to be executed within the TITAN-based device. At the moment, only the "ping" and "traceroute" commands fall into this category. You can specify a timeout (in seconds), as well as the command to be executed.

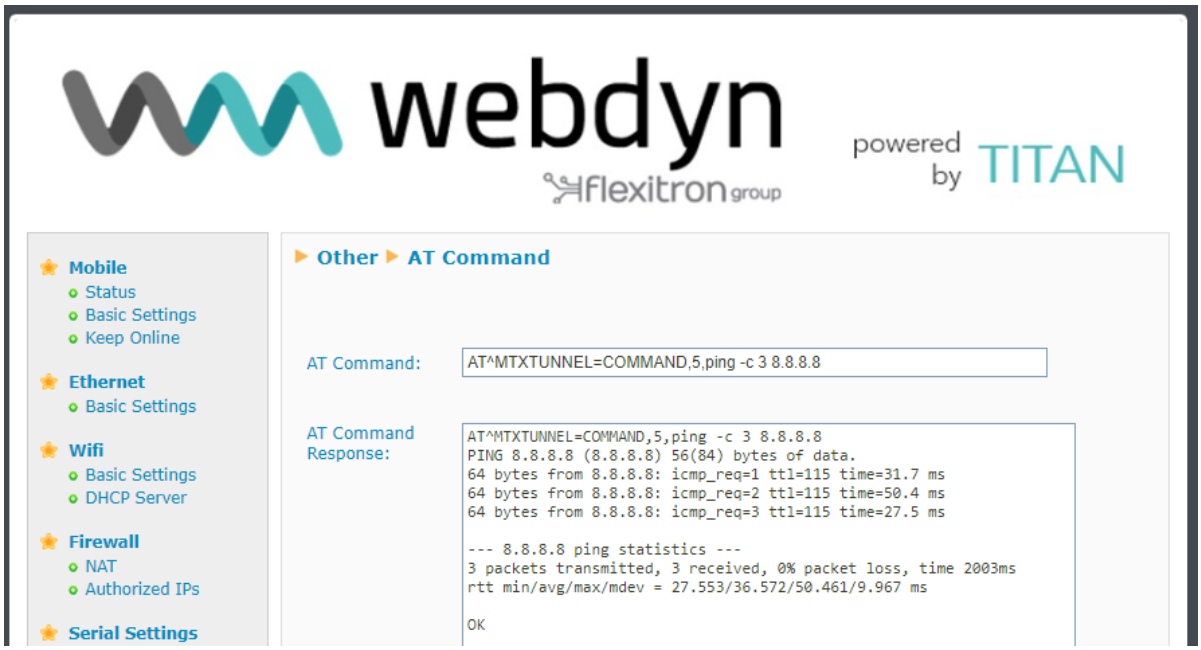
Parameters:

<timeout>: 1 … 60 seconds (timeout)

<command>: ping, traceroute

Example: AT^MTXTUNNEL=COMMAND,5,ping -c 3 8.8.8.8

(This example would execute 3 PINGs to IP 8.8.8.8 with a timeout of 5 seconds)



AT commands related to MODBUS:

- **AT^MTXTUNNEL=GETMODBUS,<modbusAddress>;<addressFirstRegister>;<numWords>;<comand>**

Action: returns the value of one of more Modbus registers for a particular device. Please note: Modbus related-parameters are separated by ";" not ","

Parameters:

<modbusAddress>: Modbus device address (1 … 255) IP address @dir:puerto

<addressFirstRegister>: address of the first register to be read (0 … 65,535)

<numWords>: number of Modbus registers to be read (1 ... 64)

<command>: Write Modbus command (5,6,15,16)

Result:

OK: Command executed correctly. This will return the words read, separated by ","

ERROR: Command executed, but with an error.

Example: AT^MTXTUNNEL=GETMODBUS,192.168.1.200:502;1;1;3;3

sent from the Web configuration environment (but it could also be sent by SMS or Remote Console (Telnet)), obtaining the values 20, 21 and 22 as responses.

- **AT^MTXTUNNEL=SETMODBUS,<modbusAddress>;<addressFirstRegister>;<command>;<data1>;<data2>;< …dataX>**

Action: sets the value of one or more Modbus registers for a particular device. Please note: Modbus related-parameters are separated by ";" not ","

Result:

OK: Command executed successfully

ERROR: Command executed, but with an error.

Parameters:

<modbusAddress>: Modbus device address (1 … 255) IP address @dir:puerto

<addressFirstRegister>: address of the first register to be written (0 … 65,535)

<command>: Write Modbus command (5,6,15,16)

-<data1>, ... <dataX>: values of Modbus registers to be written (0 ... 65535)

Example: AT^MTXTUNNEL=SETMODBUS,1;3;16;10;11;12;13;14;15

writes to the Modbus RTU device with address 1 the values 10,11,12,13,14 and 15**, starting in register** 3 **and using the Modbus write command** 16.

Example: AT^MTXTUNNEL=SETMODBUS,192.168.1.202@1:502;3;16;10;11;12;13;14;15

writes to the Modbus TCP device with IP address **192.168.1.202 the values 10,11,12,13,14 and 15,** using RTU address @1 and TCP port **502, starting in register** 3 **and using the Modbus write command** 16**.**

Example: AT^MTXTUNNEL=SETMODBUS,1;3;6;10

writes to the Modbus RTU device with address 1 the value 10**, starting in register** 3 **and using the Modbus write command** 6.

Example: AT^MTXTUNNEL=SETMODBUS,1;18;5;1

writes to the Modbus RTU device with address 1 the value 1**, in coil** 18 **and using the Modbus write command** 5.

Example: AT^MTXTUNNEL=SETMODBUS,1;25;15;1;0;1;0;1

writes to the Modbus RTU device with address 1 the coil values 1.0,1.0 and 1, from coil 25 and using the Modbus write command 15.

AT commands related to system time:

- **AT^MTXTUNNEL=GETTIME**

Action: returns the current system time in YYYYY-MM-DDTHH:NN:SSZ (UTC) format

Result:

OK: Command executed successfully
ERROR: Command executed, but with an error.

Example:

AT^MTXTUNNEL=GETTIME 2016-21-05T10:56:52Z OK

- **AT^MTXTUNNEL=SETTIME,<dateAndHour>**
  AT^MTXTUNNEL=SETTIME,<dateAndHour>

Action: displays the current time.

Parameters:

<dateAndHour> date / time in UTC format YYYYY-MM-DDTHH:NN:SSZ

Result:

OK: Command executed successfully
ERROR: Command executed, but with an error.

Example:

AT^MTXTUNNEL=SETTIME,2023-01-10T14:42:23Z OK

AT commands related to the Titan-based device configuration:

- **AT^MTXTUNNEL=GETPARAM,<paramName>**

Action: lets you read the value of any configuration parameter in the TITAN-based

device. For example, you can view the configuration for each parameter on a Web Platform or a device connected to the Ethernet or Wifi port of the TITAN-based device. If you double click on any field on any configuration page of the TITAN-based device, the name of that field will be displayed.

Parameters:

<paramName> name of the configuration parameter to be read

Result:

OK: Configuration parameter read correctly
ERROR: Configuration parameter read, but with an error

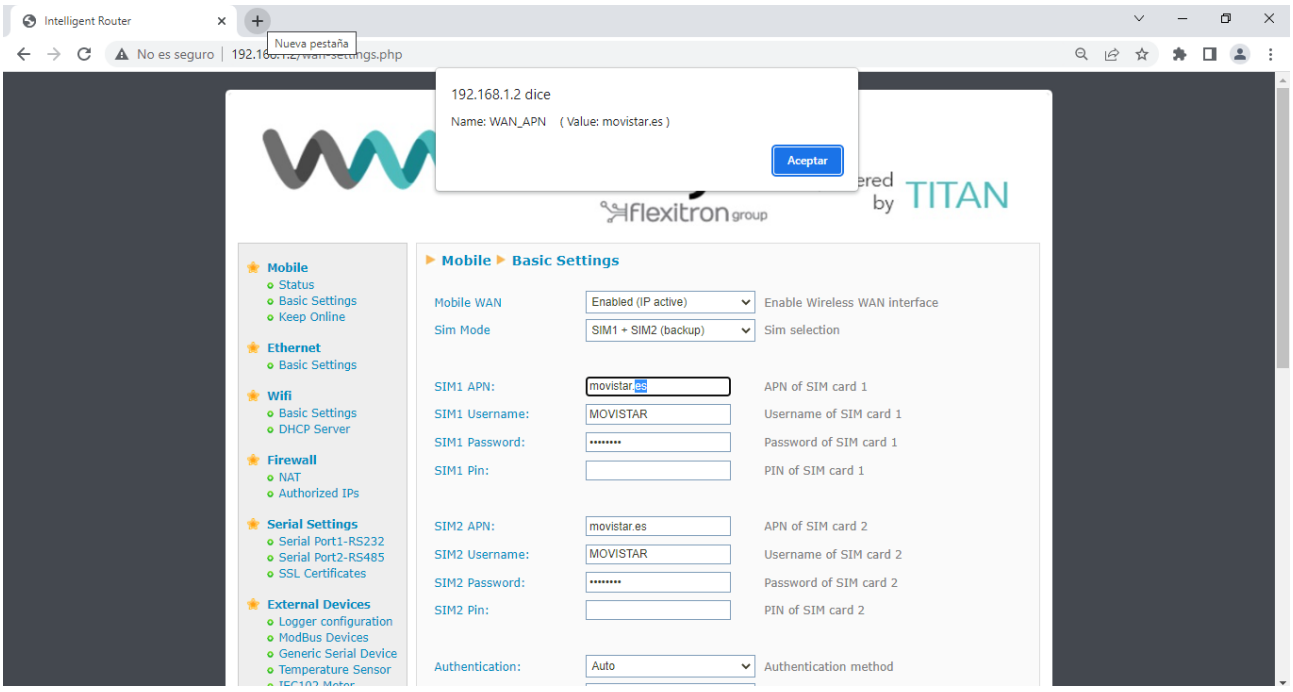
Example:

AT^MTXTUNNEL=GETPARAM,WAN_APN m2m.movistar.es OK

- **AT^MTXTUNNEL=SETPARAM,<paramName>,<paramValue>**

Action: lets you read the value of any configuration parameter in the TITAN-based device. For example, you can edit the configuration for each parameter of the device on a Web Platform or a device connected to the Ethernet or Wifi port of the TITAN-based device. If you double click on any field on any configuration page of the TITAN-based device, the name of that field will be displayed. To see the name of the field by "double-clicking", you must first send the AT command:

AT^MTXTUNNEL=SETPARAM,MTX_SHOWNAMES,1



Parameters:

<paramName> name of the configuration parameter to be modified <paramValue> New value of the configuration parameter

Result:

OK: The configuration parameter value has been edited correctly
ERROR: Configuration parameter edited, but with an error.

Example:

AT^MTXTUNNEL=SETPARAM,WAN_APN,m2m.movistar.es OK

- **AT^MTXTUNNEL=SETFILE,<fileName>,<length>,<{data_base64}>,<{data_base64}>.**

Action: in the same way that the AT^MTXTUNNEL=SETPARAM command allows you to edit the a configuration parameter value via an AT command, this parameter allows you to edit the configuration of file-type parameters (scripts, certificates, operators, ...). The name of the file will appear next to the corresponding configuration boxes.

Parameters:

<fileName> filename

<length> field data length data_base64 field, not counting { }

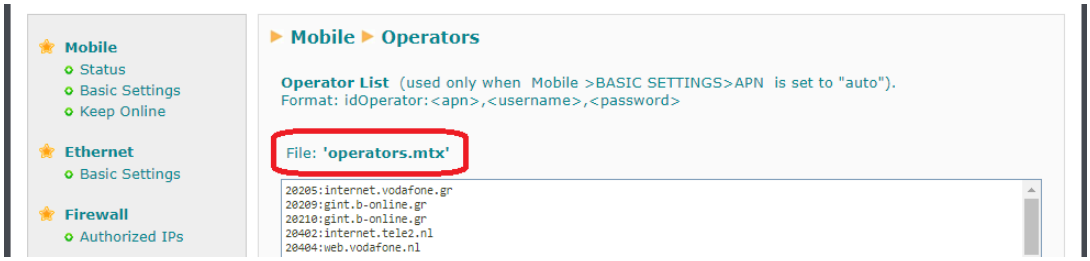
<{data_base64}>: file data encoded in base64. The text must be enclosed in { }

Result:

OK: The file was successfully uploaded
ERROR: Command executed, but with an error.

Example:

For example, if you wished to modify the "operators.mtx" file located in the Mobile → Basic Settings → Operators section,



you would need to send the following AT command:

AT^MTXTUNNEL=SETFILE,operators.mtx,220,{MjE0MDY6YWlydGVsbmV0LmVzCjIxNDA3Om1vdmlzdGFyLmVzLE1PVklTVEFSLE1PVklTVEFSCjIxNDA4OmludGVybmV0LmV1c2thbHRlbC5tb2JpCjIxNDE2OmludGVybmV0LnRlbGVjYWJsZS5lcwoyMTQxNzppbnRlcm5ldC5tdW5kby1yLmNvbQoyMTQxODppbnRlcm5ldC5vbm8uY29tCg==}

NB: here are some useful links to help you perform your testing:

https://www.base64encode.org/

https://wordcounter.net/character-count

- **AT^MTXTUNNEL=GETFILEHASH,<fileName>**

Action: returns the HASH of a file stored within the device.

Parameters:

<fileName> filename

Result:

{<HASH>}OK: File HASH.
ERROR: Command executed, but with an error.

Example:

For example, if you wish to look up the HASH of the "operators.mtx" file,

AT^MTXTUNNEL=GETFILEHASH,operators.mtx

{06FD031D0CE8B58EA41A4E282047818E0EB74EFEEF3097F719968571DD7D8BC3}

OK

AT commands related to IEC-60870-5-102 meters

- **AT^MTXTUNNEL=SETIEC102,<IDMETER>**

Action: by executing this command, you can initiate real-time value readings for meters with the IEC-60870-5-102 protocol. The service available in the "External Devices →IEC102 Meter" section must be enabled and correctly configured.

Parameters:

<IDMETER>: meter identifier. Tag configured in the "External Devices →IEC102 Meter" section or in the code of a script using the mtx function. iec102AddMeter

Result:

OK: The process has started.
ERROR: An error has occurred at the start of the BUSY process:
serial bus occupied by a CSD call, IP-RS232 gateway

Example:

AT^MTXTUNNEL=SETIEC102,ID0001 OK

- **AT^MTXTUNNEL=GETIEC102,<IDMETER>**

Action: reading the result of the AT^MTXTUNNEL=SETIEC102,<IDMETER> command execution. The service available in the "External Devices →IEC102 Meter" section must be enabled and correctly configured.

Parameters:

<IDMETER>: meter identifier. Tag configured in the "External Devices →IEC102 Meter" section or in the code of a script using the mtx function. iec102AddMeter

Result:

OK: Correct reading
ERROR: Reading error or reading process not yet complete.

Example:

AT^MTXTUNNEL=GETIEC102,ID0001
{"IMEI":"869101054255506","TYPE":"IEC102","TS":"2023-01-26T09:22:37Z","P":"MTX1","ID":"ID0001","VABA":0,"VABRI":0,"VABRC":0,"PAT":0,"PRT":0,"FPT":1000,"PAF1":0,"PRF1":0,"FPF1":1000,"PAF2":0,"PRF2":0,"FPF2":1000,"PAF3":0,"PRF3":0,"FPF3":1000,"IF1":0,"TF1":1156,"IF2":0,"TF2":1141,"IF3":0,"TF3":1}

OK

- **AT^MTXTUNNEL=SETIEC102_CTAVM2, <IDMETER>,<hourIni>,<minuteIni>,<dayIni>,<monthIni>,<yearIni>,<hourEnd>,<minuteEnd>,<dayEnd>,<monthEnd>,<yearEnd>,<yearEnd>.**

Action: by executing this command, you can initiate fiscal close (contract I) readings for meters with the IEC-60870-5-102 protocol. The service available in the "External Devices →IEC102 Meter" section must be enabled and correctly configured.

Parameters:

<IDMETER>: meter identifier. Tag configured in the "External Devices →IEC102 Meter" section or in the script code using the mtx. iec102AddMeter<hourIni> function: interval start time (0 ... 23) <minuteIni>: initial minute of the interval (0 ... 59) <dayIni>: initial day of the interval (1 ... 31) <monthIni>: starting month of the interval (1 ... 12) <yearIni>: starting year of the interval (20 ... 99) <hourEnd>: Interval end time (0 ... 23) <minuteEnd>: end minute of the interval (0 ... 59) <dayEnd>: end day of the interval (1 ... 31) <monthEnd>: end month of the interval (1 ... 12) <yearEnd>: end year of the interval (20 ... 99)

Result:

OK: The process has started.
ERROR: An error has occurred at the start of the BUSY process:
serial bus occupied by a CSD call, IP-RS232 gateway

Example:

AT^MTXTUNNEL=SETIEC102_CTAVM2,ID001,0,0,22,12,22,0,0,24,1,23 OK

- **AT^MTXTUNNEL=GETIEC102_CTAVM2,<IDMETER>**

Action: reading the result of the AT^MTXTUNNEL=SETIEC102_CTAVM2,<IDMETER> command execution. The service available in the "External Devices →IEC102 Meter" section must be enabled and correctly configured.

Parameters:

<IDMETER>: meter identifier. Tag configured in the "External Devices →IEC102 Meter" section or in the code of a script using the mtx function. iec102AddMeter

Result:

OK: Correct reading
ERROR: Reading error or reading process not yet complete.

Example:

AT^MTXTUNNEL=GETIEC102_CTAVM2,ID001
{"IMEI":"869101054255506","TYPE":"IEC102_CTAVM2","TS":"2023-01-26T09:41:29Z","P":"MTX1","ID":"ID001","CTAVM2":[{"DO":20,"EaA":0,"EiA":0,"CA":2,"EaRi":0,"EiRi":0,"CRi":2,"EaRc":0,"EiRc":0,"CRc":2,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":2,"EPA":0,"CE":128,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},{"DO":21,"EaA":0,"EiA":0,"CA":2,"EaRi":0,"EiRi":0,"CRi":2,"EaRc":0,"EiRc":0,"CRc":2,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":2,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},{"DO":22,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},{"DO":23,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},{"DO":24,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},{"DO":25,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-

0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"},{"DO":26,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2023-01-09T12:58-0","CMA":0,"EPA":0,"CE":0,"DINI":"2022-12-01T00:00-0","DEND":"2023-01-09T12:58-0"}]}
OK

- **AT^MTXTUNNEL=SETIEC102_CCINX2,<IDMETER>,<hourIni>,<minuteIni>,<dayIni>,<monthIni>,<yearIni>,<hourEnd>,<dayEnd>,<monthEnd>,<yearEnd>,<yearEnd>,<hourEnd>.,<minuteEnd>,<dayEnd>,<monthEnd>,<yearEnd>,<absoluteIncremental>,<period>.**

  Action: by executing this command, you can initiate integrated total value readings for meters with the IEC-60870-5-102 protocol. The service available in the "External Devices →IEC102 Meter" section must be enabled and correctly configured.

  Parameters:

  <IDMETER>: meter identifier. Tag configured in the "External Devices →IEC102 Meter" section or in the script code using the mtx. iec102AddMeter<hourIni> function: interval start time (0 ... 23) <minuteIni>: initial minute of the interval (0 ... 59) <dayIni>: initial day of the interval (1 ... 31) <monthIni>: starting month of the interval (1 ... 12) <yearIni>: starting year of the interval (20 ... 99) <hourEnd>: Interval end time (0 ... 23) <minuteEnd>: end minute of the interval (0 ... 59) <dayEnd>: end day of the interval (1 ... 31) <monthEnd>: end month of the interval (1 ... 12) <yearEnd>: end year of the interval (20 ... 99) <absoluteIncremental>: 0=absolute, 1=incremental <period>: 0=hourly, 1=every quarter of an hour

  Result:

  OK: The process has started.
  ERROR: An error has occurred at the start of the BUSY process:
  serial bus occupied by a CSD call, IP-RS232 gateway

  Example:

  AT^MTXTUNNEL=SETIEC102_CCINX2,ID001,15,0,19,1,23,20,0,19,1,23,1,0 OK

- **AT^MTXTUNNEL=GETIEC102_CCINX2,<IDMETER>**

  Action: reading the result of the AT^MTXTUNNEL=SETIEC102_CCINX2,<IDMETER>

command execution. The service available in the "External Devices →IEC102 Meter" section must be enabled and correctly configured.

Parameters:

<IDMETER>: meter identifier. Tag configured in the "External Devices →IEC102 Meter" section or in the code of a script using the mtx function. iec102AddMeter

Result:

OK: Correct reading
ERROR: Reading error or reading process not yet complete.

Example:

AT^MTXTUNNEL=GETIEC102_CCINX2,ID001

{"IMEI":"869101054255506","TYPE":"IEC102_CCINX2","TS":"2023-01-26T09:54:10Z","P":"MTX1","ID":"ID001","CCINX2":[{"DO":11,"TI1":0,"TI2":0,"TI3":0,"TI4":0,"TI5":0,"TI6":0,"TI7":0,"TI8":0,"Q1":0,"Q2":0,"Q3":0,"Q4":0,"Q5":0,"Q6":0,"Q7":128,"Q8":128,"DATE":"2023-01-19T15:00-0"},{"DO":11,"TI1":0,"TI2":0,"TI3":0,"TI4":0,"TI5":0,"TI6":0,"TI7":0,"TI8":0,"Q1":0,"Q2":0,"Q3":0,"Q4":0,"Q5":0,"Q6":0,"Q7":128,"Q8":128,"DATE":"2023-01-19T16:00-0"},{"DO":11,"TI1":0,"TI2":0,"TI3":0,"TI4":0,"TI5":0,"TI6":0,"TI7":0,"TI8":0,"Q1":0,"Q2":0,"Q3":0,"Q4":0,"Q5":0,"Q6":0,"Q7":128,"Q8":128,"DATE":"2023-01-19T17:00-0"},{"DO":11,"TI1":0,"TI2":0,"TI3":0,"TI4":0,"TI5":0,"TI6":0,"TI7":0,"TI8":0,"Q1":0,"Q2":0,"Q3":0,"Q4":0,"Q5":0,"Q6":0,"Q7":128,"Q8":128,"DATE":"2023-01-19T18:00-0"},{"DO":11,"TI1":0,"TI2":0,"TI3":0,"TI4":0,"TI5":0,"TI6":0,"TI7":0,"TI8":0,"Q1":0,"Q2":0,"Q3":0,"Q4":0,"Q5":0,"Q6":0,"Q7":128,"Q8":128,"DATE":"2023-01-19T19:00-0"},{"DO":11,"TI1":0,"TI2":0,"TI3":0,"TI4":0,"TI5":0,"TI6":0,"TI7":0,"TI8":0,"Q1":0,"Q2":0,"Q3":0,"Q4":0,"Q5":0,"Q6":0,"Q7":128,"Q8":128,"DATE":"2023-01-19T20:00-0"}]}

OK

DO = object address

TIx, where x=1...8, the integrated totals, where:

Dirección	Objeto de Información
1	Totales Integrados de Activa Entrante.
2	Totales Integrados de Activa Saliente
3	Totales Integrados de Reactiva primer cuadrante
4	Totales Integrados de Reactiva segundo cuadrante
5	Totales Integrados de Reactiva tercer cuadrante
6	Totales Integrados de Reactiva cuarto cuadrante
7	Datos de reserva 1
8	Datos de reserva 2

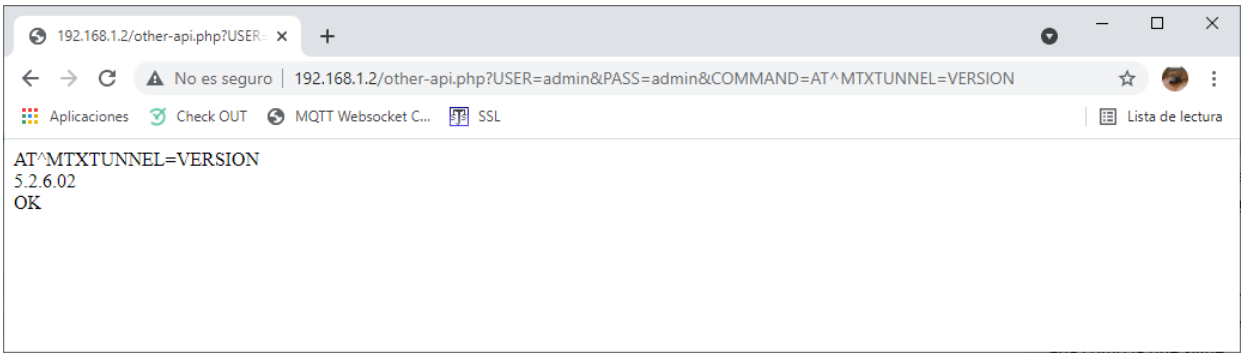
Qx, where x=1...8, the qualifiers, where:

Bit	Iden.	Descripción
7	IV	La lectura es válida (IV=0)
6	CA	Contador sincronizado durante el período (CA=1)
5	CY	Overflow (CY=1)
4	VH	Verificación horaria durante el período (VH=1)
3	MP	Modificación de parámetros durante el período (MP=1)
2	INT	Se produjo un intrusismo durante el período (INT=1)
1	AL	Período incompleto por fallo de alimentación en el período (AL=1)
0	RES	Reserva. Nótese su empleo en Tarificación.

For the rest of the standard commands, please refer to the AT commands manual for the internal gsm module of the TITAN-based for information on the exact format and functionality of each of the module's commands. If you do not have this documentation, please send an email to contact@webdyn.com

In addition to sending AT commands via SMS, Telnet, Modbus TCP, etc., it is possible to send AT commands via HTTP GET, both locally and remotely. For example, to execute a temperature reading command on the GSM module, you would need to make a call like this:

http://192.168.1.2/other-api.php?USER=admin&PASS=admin&COMMAND=AT^MTXTUNNEL=VERSION



6.- New Firmware releases

6.**17** - Potential autonomous readings of load curves for IEC-60870-5-102 meters from a script. New SCRIPT functions: mtx.iec102SetIntegratedTotal y mtx. iec102GetIntegratedTotal.

- New SCRIPT in relation to example 52, showing how to read the load curves of an IEC-60870-5-102 meter.

- New commands – AT^MTXTUNNEL=SETIEC102_CCINX2 and AT^MTXTUNNEL = GETIEC102_CCINX2 – to read load curves for IEC-60870-5-102 meters

6.**18** - Improved handling of TCP Client connections in the event of context loss.

6.**19.-** MAC filtering from the "Firewall → Mac Filter" section

- IEC60870-5.102. Added parameters (patDir, paf1Dir, paf2Dir, paf3Dir) to specify imported/exported energy in the JSON of instant energy values.

- IEC60870-5.102. New configuration parameter in the "External Devices → Meter 102" section, which allows you to assign designations to the pat/fpt, paf1/fpf1, paf2/fpf2 and paf3/fpf3 parameters.

- Aesthetic enhancements to the Titan Scripts programming display.

- Added HTTP PUT and HTTPS PUT method capabilities in the LOGGER and SCRIPTS sending options.

- SCRIPT in example 39 modified, including the HTTP PUT method.

- New mtx.loggerSendingEnabled function for the SCRIPTS section. This function will allow you to enable/disable the service that sends the data stored in the LOGGER.

- A new SCRIPT to the one in example 7b. Reading a Modbus TCP register.

- A new SCRIPT to the one in example 14b. Reading Modbus RTU registers, customising the JSON to be sent, storing data in the LOGGER and scheduling sending within a specific time schedule.

- New AT^MTXTUNNEL=GETFILEHASH,<filename> command, which allows you to read the HASH of an internal configuration file.

6.20.- Issue of not being able to send special characters by SMS solved ^ _

- Improved management of SMSs sent from scripts

- In the "ID String" field of the client IP-Serial gateways, it is possible to specify that the tags [IMEI] , [CR] and [LF] be automatically replaced by the IMEI and the special characters 0x13 and 0x10.

- New functions for scripts: mtx.convert2WordToUnsignedInt32 , mtx.convert2WordToSignedInt32 , mtx.convert2WordToFloat32 , mtx.convert4WordToSignedInt64 useful for modbus data conversions

- Added 0x06 modbus write command

- New functions for the mtx.configParamGet and mtx.configParamSet scripts that allow the router configuration to be read and changed from the scripts.

- Added example scripts 53 and 54 for DLMS meter reading

6.21.- Option to edit Modbus devices registered in the section "External Devices → Modbus Devices"

- Support for the new WEBDYN-EXPERT-ROUTER device

6.22.- Support for autonomous meter reading with the IEC-60870-5-102 protocol through an optical probe interface.

- Support for the new WEBDYN-EASY-ROUTER device

6.24.- New OTAP firmware update feature.

- New configuration options for the SIM SWAP feature for DUAL SIM devices.

- Automatic logoff option with the IEC-60870-5-102 protocol for TCP Server - RS232/RS485 gateways

6.25.- - New VPN “ZeroTier” feature. New AN71 application note.

- New “Modbus Expert” reading feature for Modbus devices. New AN72 application note.

- New PC-based configurations feature for mass production. New AN73 application note.

SALES CONTACT

SPAIN

C/ Alejandro Sánchez 109
28019 Madrid

Phone no. 1: 902.19. 81.46
Phone no. 2: +34 91 560 27 37
Email: contact@webdyn.com

FRANCE

26 Rue des Gaudines
78100 Saint-Germain-en-Laye

Phone: +33.139042940
Email: contact@webdyn.com

INDIA

803-804 8th floor, Vishwadeep Building
District Centre, Janakpurt, 110058 New Delhi

Phone: +91.1141519011
Email: contact@webdyn.com

PORTUGAL

LusoMatrix Lda.
Av. Coronel Eduardo Galhardo 7-1ºC
1170-105 Lisboa, Portugal

Phone: +351.218162625
Email: comercial@lusomatrix.pt

APAC

9F, No. 156, Sec. 3, Minsheng E. Rd.
Songshan Dist., Taipei City 10596, Taiwan

Phone: +886.965333367
Email: contact@webdyn.com

SUPPORT

Madrid Offices

Phone: +34.915602737

Email: iotsupport@matrix.es

Saint-Germain-en-Laye Offices

Phone: +33.139042940

Email: support@webdyn.com

Delhi Offices

Phone: +91.1141519011

Email: support-india@webdyn.com

Taipei City Offices

Phone: +886.905655535

Email: iotsupport@matrix.es