WEBDYN

GATEWAY SPECIFICATION

WebdynRF Wavenis

Gateway dedicated to energy control and Smart Metering

INDEX

1 Introduction

The WebdynRF gateway is a wireless Ethernet/GPRS/3G gateway.
It can be equipped (manufacturing option) with any of the following wireless transceivers:

- Wavenis 25mW or 500mW
- Wireless M-Bus (868MHz or 169MHz)
- ELA active RFID

With a Wavenis radio, it can manage up to 5000 Wavenis modules. Any alarm from a module is sent to the remote server. It periodically collects data from these modules and uploads them periodically.
With a Wireless M-Bus radio, it can collect data in S1, T1 or N1 mode. It supports OMA encryption.
With an ELA active RFID radio, it can monitor the presence of active RFID tags and collect data such as temperature and humidity.

2 Features

2.1 Overview

The WebdynRF gateway is powered by an ARM 9 CPU equipped with 1Gbit of DDR2 volatile memory and 1Gbit of NAND Flash non-volatile memory. It is running the Linux operating system.

2.2 Cellular connectivity

The cellular connectivity is either 2G or 3G (manufacturing option).

- The 2G modem is a quad-band GPRS class 10 onboard module (BGS2 from Cinterion).
- The 3G modem is a 900 MHz UMTS/HSDPA daughter-board module (EU3 from Cinterion).

In both cases, the WAN interface has the following characteristics:

- Its power supply is driven by the processor. This reduces the overall power consumption of the gateway and enable a complete reset of the modem if necessary.
- Two types of SIM are supported (manufacturing option):
    - Soldered SIM (MFF2, VQFN8)
    - Standard Push-Push SIM slot (2FF).

2.3 Wavenis

The gateway is equipped with a Wavecard module, 25mW or 500mW (manufacturing option). In addition the board is ready to host an Excelyo Wavenis module.
To take into account the coexistence of the Wavenis and cellular radios, a SAW filter is added on the Wavenis RF path and an optional shield isolate the Wavenis radio.

2.4 Interfaces

Four LEDs (power, CPU, WAN, Data) indicates:

- The availability of the external power supply,
- The processor activity,
- Communication over the cellular network,
- Communication over the Wavenis radio.

The following logical I/Os are available:

- 1 pin-hole hardware reset button
- 1 push button
- 3 logical inputs
- 1 relay output (dry contact)

Three additional inputs are dedicated to the UPS extension:

- UPS power availability
- UPS battery in charge
- UPS battery default detection

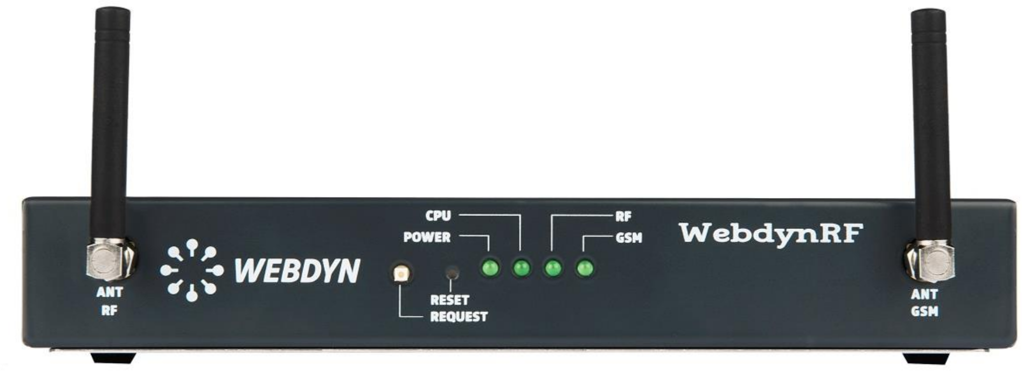
The following communication ports are available:

- RS232
- RS485
- USB
- Ethernet (100 Mbit/s)

2.5 Power supply

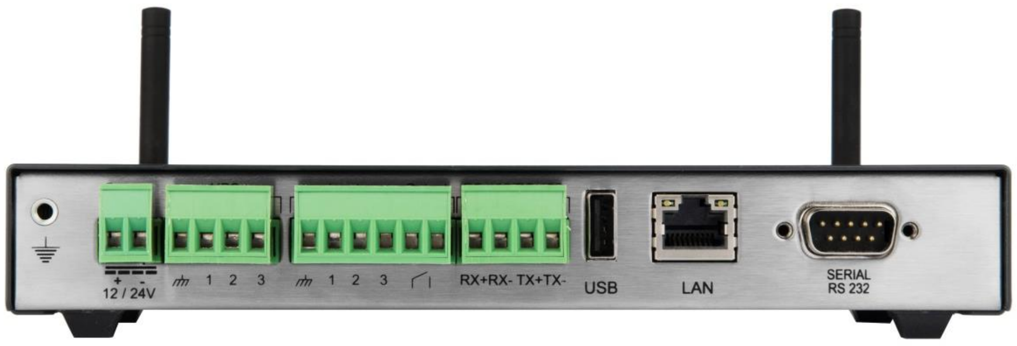
The gateway must be connected to a DC power supply between 10 volts and 30 volts.
It contains an internal low-capacity battery (type Lithium-Ion). This battery allows the gateway to inform the remote server of any loss of power supply. This internal battery is managed directly by the processor power management unit.
A DC/DC converter is dedicated to the modem and the radio in order to sustain the peak courant. Moreover the processor drives both power-supplies independently. This permits to reduce the overall power consumption and enable a hardware reset of both radios if necessary.
It can be connected to an external UPS bloc to manage an external high capacity battery and a photovoltaic power supply. The gateway has dedicated I/O to monitor the external UPS module.

2.6 Form factor



The front face includes:

- 1 SMA connector for the Wavenis antenna
- 1 push button
- 1 reset button (pin-hole type)
- 4 LEDs
- 1 SIM slot (manufacturing option)
- 1 SMA connector for the GPRS/3G antenna



The rear face includes:

- 1 DC power supply connector
- 1 UPS extension connector
- 1 input/output connector
- 1 RS485 connector
- 1 USB connector (manufacturing option)
- 1 Ethernet connector
- 1 RS232 connector

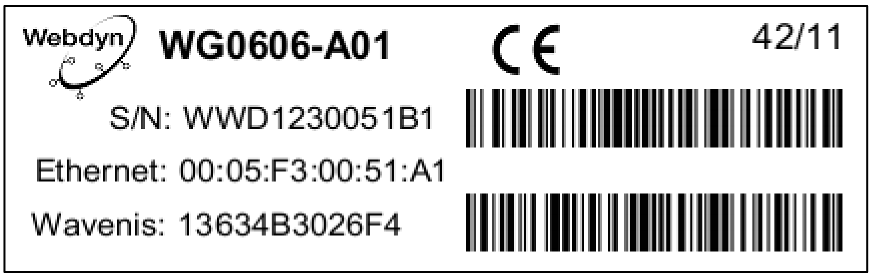
The gateway can rest on a shelf on its rubber feet. An optional fixation kit allows the gateway to be either mounted on a Rail DIN, or mounted directly on a wall with screws.

The size of the gateway is 200x110x31.5 mm.

2.7 Label

A label, placed on the bottom of the gateway, contains the following information:

- Model number
- Manufacturing batch reference
- Serial number (in both text format and barcode)
- Ethernet MAC address
- Wavenis Address (in both text format and barcode)



Webdyn WG0606-A01 CE 42/11

S/N: WWD1230051B1

Ethernet: 00:05:F3:00:51:A1

Wavenis: 13634B3026F4

2.8 Certifications

- CE
- R&TTE
- RoHS

3 Configuration

3.1 Parameters

The gateway parameters are handled in a structured manner. The configuration can be exported as an XML file. The installation of a new configuration and the modification of the current configuration are done using an XML file with the same format. This format is specified by an XML schema (see Appendix A).

The XML schemas specifying the format of the various XML files used by the gateway may evolve in future versions as features are added. These changes will be made so that old XML files remains valid with the new XML schemas. Moreover as XML files that are generated by the gateway may contain additional elements, their processing should be implemented so that new elements are ignored.

The main parameters of the gateway are listed below:

Name	Values	Description
/uid		Unique gateway identifier (by default, the last 6 digits of the Ethernet MAC address).
/name		Name of the gateway (for informative purpose only). By default, the name is made of the MAC address prefixed by « WGRF_ ».
/enable_local_config	false, **true**	Enable local configuration.
/com/modem/pin/mode	**off**, manual, automatic	SIM PIN code mode (see §9.1)
/com/modem/pin/code	0000	PIN code
/com/modem/call_number		Call number
/com/modem/apn		APN
/com/modem/login		Login
/com/modem/password		Password
/com/modem/mode	**ondemand,** alwayson, alwaysoff, off	See the mode description in paragraph §9.1).
/com/modem/delay	**60**	Delay before disconnection in ondemand mode (seconds).
/com/modem/whitelist/caller_id		Whitelist of caller IDs for connection request and incoming SMS.
/com/ethernet/use_dhcp	**false**, true	DHCP client.
/com/ethernet/ip	192.168.1.12	IP
/com/ethernet/netmask	255.255.255.0	IP network mask.
/com/ethernet/gateway		IP network gateway.
/com/ethernet/dns/server		DNS server addresses.
/com/keepalive/method	icmp, tcp, **off**	Keepalive method.
/com/keepalive/address		Destination of keepalive queries.
/com/keepalive/port	**5000**	Destination port of keepalive queries (TCP method only).
/com/keepalive/period	**600**	Keepalive queries periodicity in seconds.
/com/keepalive/timeout	**30**	Keepalive queries timeout in seconds.

/com/request/upload	false, **true**	Connect to the remote server on button request
/com/request/include_status	false, **true**	Upload status on button request
/com/request/sms_status_recipient		Recipient for the status SMS on button request.
/com/time/ntp/server		NTP server addresses.
/com/time/timezone		Local timezone (using standard zoneinfo name such as “Europe/Paris”).
/com/time/alarm_threshold	**0**	Alarm threshold (seconds, 0=off).
/com/ftp/address		FTP server address.
/com/ftp/login		FTP login
/com/ftp/password		FTP password
/com/ftp/mode	**passive**, active	FTP mode
/com/ftp/secured	**false**, true	Use secure FTP protocol.
/com/ftp/trust_mode	trust_peer, **verify_peer**	Check peer certificate validity against CA certificates.
/com/ftp/root_path	/	Root path on the FTP server.
/com/ftp/ws_notification	**none**, put, get, both	Web service notification of FTP file download/upload.
/com/ws/address		Web service address.
/com/ws/login		WS login
/com/ws/password		WS password
/com/ws/secured	**false**, true	Use SSL/TLS (HTTPS) for WS.
/upload/config/method	**ftp**|ws|none	Method used to upload configuration.
/upload/config/omit_password	**false**, true	Do not include passwords in uploaded configuration files.
/upload/supervision/method	**ftp**|ws	Method used to upload supervision data.
/upload/alarm/method	**ftp**|ws	Method used to upload alarms.
/upload/data/method	**ftp**|ws	Method used to upload data.
/upload/data/format	xml|**csv**	Format used to upload data.
/upload/data/schedule		Data upload schedule ID.
/upload/common/size_limit	10	Maximum size of uploaded files (in MB, unlimited if not defined or set to 0)
/alarm/*		Alarm engine configuration (described in paragraph 0).
/scheduler/*		Scheduler configuration (described in paragraph 10).
/wavenis/*		Wavenis configuration (described in paragraph 4.3).
/metering/*		Metering configuration (described in

		paragraph 5).
/rfid/*		Active RFID configuration (described in paragraph 6).
/modbus/*		Modbus configuration (described in §7).
/system/log/level	**5**	Log level (see paragraph 12).
/system/password/admin	**high**	Passwords for local HTTP and FTP services.
/system/password/install	**medium**	
/system/password/data	**low**	
/system/ports/*		See paragraph 8.2.

The configuration can be modified locally or remotely. Any modification triggers the upload of the new configuration to the server.
/com/modem/whitelist

If the list is empty, all numbers are considered valid.

/com/modem/whitelist/caller_id
/com/request/sms_status_recipient

The phone numbers must be in their international format.
They must start with + and the country code.

3.2 Configuration by SMS

The initial configuration of the gateway can be done by SMS. Typically the connection parameters can be sent by SMS. Once this initial configuration completed, the remote server can continue the configuration of the gateway.
The first line of the SMS must contain the command « CMD=config ».
The following lines must have the format « SHORTNAME=VALUE ». The short name is made of the first letter of each element composing the parameter name:
For example, the short name for « `/`**`c`**`om/`**`m`**`odem/`**`l`**`ogin` » is CML.
The SMS content is subject to the following rules:

- The space characters at the end of a line are ignored.
- Short names are not case sensitive.
- Boolean values (false and true) can be replaced respectively by 0 and 1.
- The carriage return can be replaced by a semicolon but both can't be mixed and any semicolon presents in a value must be escaped by another semicolon.

The SMS length is limited to 160 characters.
Only the main parameters can be modified by SMS (/uid, /name, /enable_local_config and /com/*).

Example:
To do the initial configuration for the first time of a new gateway with the following context:

- Call number *99***1#
- APN "m2minternet" not requiring a login/password
- FTP-based communication with the remote server (168.112.23.123) in passive mode

You can send the following SMS:

```
CMD=config
CMC=*99***1#
CMA=m2minternet
CFA=168.112.23.123
CFL=login
CFP=password
```

Observe that all parameters using their default value have been omitted.
On reception of this SMS, the gateway will apply the parameters and connect to the remote server to upload the resulting configuration file. From there the gateway can be further configured remotely as described below.
The pairs for /com/modem/whitelist/caller_id (and /com/time/ntp/server) can be repeated. If it appears at least one, the current list of caller_id (or ntp servers) will be replaced. If it appears only once and without value, the current list is cleared.
Examples:

```
CMD=config
CTNS=1.2.3.4
```

After processing this SMS, the gateway will use the DNS server 1.2.3.4.

```
CMD=config
CTNS=1.2.3.4
CTNS=1.2.3.5
```

After processing this SMS, the gateway will use the 2 DNS servers 1.2.3.4 and 1.2.3.5.

```
CMD=config
CTNS=
```

After processing this SMS, the gateway will not use any DNS server.

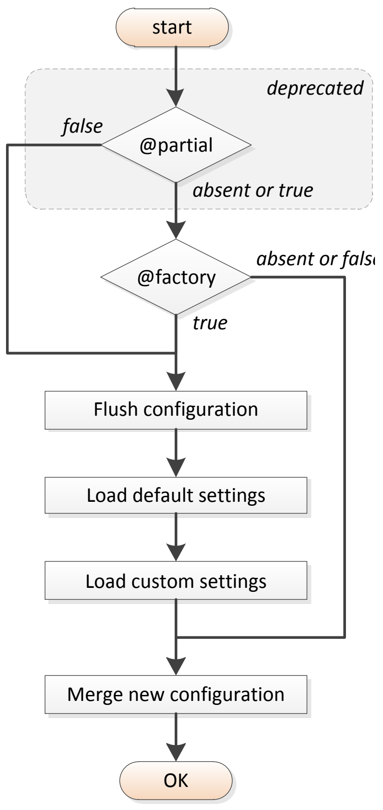
3.3 Local configuration

The gateway can be configured locally thru a web-based interface. This is possible only when /enable_local_config is true. This parameter, when false, ensures that the remote server controls the gateway configuration.

3.4 Remote configuration

The supervision server can modify the configuration by placing an XML configuration file in the INBOX directory on the FTP server or by serving it thru the web service. The same XML format is used in both cases.
The XML file is processed as a new configuration if the XML attribute “factory” is present and has the value “true”.

The attribute partial is still supported but deprecated. The factory attribute set to true is equivalent to the attribute partial set to false.



When the factory attribute is not present or set to false, the value of the configuration parameters present in the new configuration file are updated.
When a list is present in the new configuration file, the whole list is replaced. This is in particular the case for the list of wavenis modules and schedules.
For example if /config/wavenis/modules is present in the configuration file, its content will replace the previously configured list of modules.

4 Wavenis

4.1 Transparent mode

The gateway features a transparent TCP/Wavenis mode. This mode allows the gateway to be used as a virtual waveport on a PC locally connected to the gateway thru Ethernet. This mode is not intended to be used remotely.
This mode can be deactivated by setting to false the parameter `/wavenis/bridge/enabled`.
When this mode is enabled, as soon as a TCP connection is established on the dedicated TCP port (`/wavenis/bridge/port`), the gateway stops using the wavecard for data collection purpose and reroutes all Wavenis communication between the wavecard and the TCP connection.
A software for Microsoft Windows is provided. This software will establish the TCP connection with the gateway and will create a virtual serial port. It is then possible to use the gateway as a virtual waveport with Coronis utility tools.

While the transparent mode is in use, all Wavenis data received will be forwarded to the TCP client, including possible alarms from a Wavenis module.

4.2 Supported Wavenis modules

The gateway supports the following Wavenis modules:

- Waveflow (1, 2 and 4 inputs)
- Wavetherm Dallas (1 and 2 inputs), PT100 (1 input) and PT1000 (1 input)
- Wavesense 4-20mA (1 input) and 0-5V (1 input)
- Wavelog (2 and 4 inputs)
- Wavetic (1 and 2 inputs)

The gateway does not manage the configuration of these modules.

4.3 Wavenis configuration

Name	Values	Description
/wavenis/bridge/enabled	**false**, true	
/wavenis/bridge/port	**4000**	TCP port used for the Wavecard/Ethenet bridge.
/wavenis/time/mode	utc,local,**nodst**	Time mode (see §4.5)
/wavenis/alarm/mode	**basic,** extended	Alarm processing mode
/wavenis/alarm/sources/unknown	on, **off**, delayed	Alarm when receiving data from an unknown module
/wavenis/alarm/sources/route	**on**, off, delayed	Alarm when receiving data from a known module but not following the correct routing
/wavenis/modules/*		Ordered list of Wavenis modules.

Each module is configured as follow:

Name	Description
module/address	Wavenis address
module/label	Informative-only module's name
module/type	Module's type
module/repeaters	List of repeaters
module/mode	Data mode (immediate, datalog).
module/nbinput	Number of indexes.
module/schedule	Schedule ID for data collection

Any Wavenis address can be given in its hexadecimal form (12 digits) or in its decimal form (15 digits, with optional dash after the 5th and 7th digits).

The addresses of the repeaters must be given in the gateway to module order.
The data is requested from all modules associated with a given schedule in the order configured. A query to a given module is repeated up to 3 times if necessary.

4.4 Wavenis alarms

When the gateway receives a Wavenis alarm message, it acknowledges it (to the sender).
If the alarm comes from a known module, it processes it then initiates a connection to the remote server to upload it.
If the alarm comes from an unknown module, an alarm“wavenis_unknown” is triggered. This alarm will be sent only once per module.
If the alarm comes from a known module but did not follow the correct route (parameter `/wavenis/alarm/sources/route`), an alarm “wavenis_route” is triggered. This alarm will be sent only once per module. In this case, the original wavenis alarm will still be processed as described below.
The gateway does not filter the Wavenis alarms (i.e. there are no settings to select the alarms from a given modules). They can be activated and/or deactivated on the module side.
Two modes of alarm processing are available: basic and extended (parameter `/wavenis/alarm/mode`).

- In the basic mode, the information sent to the server is solely based on the wavenis alarm status frame.
- In the extended mode, the gateway will query the wavenis module to retrieve more information about the alarm.

In both case the alarm is uploaded to the remote server in XML as specified by the alarm XML schema (see Appendix C).

The XML format used to report alarms in short mode is similar but not identical to the format used by the WGE-G-COR (Wavegate310).

4.5 Wavenis time management

The gateway is configured to a given timezone which may include Daylight saving time (DST) (typically 1 hour extra shift from GMT/UTC time during the summer). All data handled by the gateway is time-stamped with its local time.
Wavenis modules time-stamp data using their own clock (RTC). Since Wavenis modules do not handle DST, it is necessary to indicate to the gateway how the RTC of the Wavenis modules is to be handled.

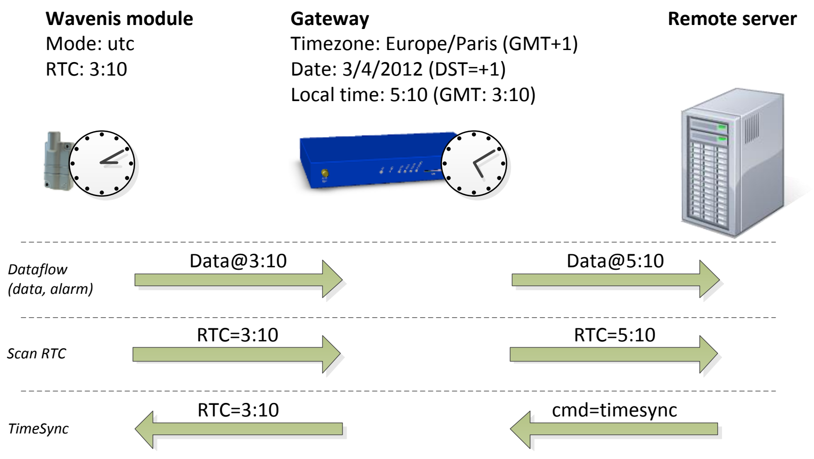
The parameter /wavenis/time/mode can have the following value:

- utc: The RTC of the Wavenis modules are set to the UTC/GMT (Coordinated Universal Time, Greenwich Mean Time) time.
- local: The RTC of the Wavenis modules are set on the same timezone as the gateway taking into account the Daylight Saving Time.
- nodst: The Wavenis module RTC are set on the same timezone as the gateway without the Daylight Saving Time. This is the default value.

The “local” mode implies that the Wavenis modules’ RTC must be adjusted after each DST change (typically twice a year). This can be done with the timesync command. This is the mode to be used if the Wavenis datalog schedule must be executed with respect to the local time. NOTE that in this mode there will be an offset error in data generated between a DST change and the associated call to timesync.

In any cases, the gateway will convert all timestamps contained in the Wavenis data to its local time (datalog timestamps, alarm timestamps, rtc readings). The gateway will also take into account the time mode when setting the RTC of a Wavenis module.

To illustrate this, let’s consider a case where the mode is set to utc:



4.6 Wavetic support

The configuration of a Wavetic module requires the same parameters as the other types of Wavenis modules. But unlike for the other types, the gateway relies on additional parameters that are obtained directly from the module. Moreover unlike for the other types, in datalog mode, only new datalog entries will be retrieved from the module. The gateway keeps track of the last datalog entry obtained from each module.

When a new module is configured, the gateway will query the module to obtain this information. This is accomplished by triggering a “scan mode=tic” command (see §0). The result of this command will be sent to the remote server.

The “scan mode=tic” command will retrieve the following information:

- The selected TIC profile
- The profile details the custom profile is selected
- The value of static TIC labels such as ADCO (meter serial number)

This command will also reset the datalog starting point.

If the TIC profile of a configured module is modified afterward, it is necessary to use the “scan mode=tic” command. Until the completion of this command, the gateway will detect the inconsistency and return an “err_config” error (see next paragraph).

4.7 Wavenis communication error

During the data collection process, if a module is not responding, the data XML file will contain an entry with an “err_status” element indicating the origin of the error, it can be:

err_status	Description
none	No error (omitted)
no_response	The module did not respond
err_repeater_ 1	The first repeater did not respond
err_repeater_ 2	The second repeater did not respond
err_repeater_ 3	The third repeater did not respond
err_config	The response is not coherent (WaveTIC)

Moreover the entry will contain an “retry_count” element containing the number of retry if at least one retry was necessary.

5 Metering

5.1 Pulse

The digital input can be selectively configured as pulse counter (see §8.2).
Once a digital input set in pulse mode, an associated counter will be incremented after each pulse lasting more than 10ms. The current value will be logged for each occurrence of the specified schedule. The following parameters in `/config/metering/pulse` are available:

Name	Description
schedule	Schedule ID for pulse data collection
input_1/label	Informative-only input name
input_1/unit	Informative-only pulse unit (and weight)
input_2/label	Informative-only input name
input_2/unit	Informative-only pulse unit (and weight)
input_3/label	Informative-only input name
input_3/unit	Informative-only pulse unit (and weight)

The label and unit parameters are added to the logged data along the index value.

5.2 Wired M-Bus

Data can be collected from M-Bus devices. An M-Bus transceiver must be connected to the RS232 port and the RS232 must be configured in M-Bus mode (see §8.2).
The M-Bus devices must be configured to have a unique primary address on the bus.
A bus scan must be initiated from the web interface. The M-Bus devices discovered during this scan will then be polled at each occurrence of the specified schedule. If devices are added or removed from the bus, a new scan must be initiated in order to take into account the modification.
The following parameter in `/config/metering/mbus` is available:

Name	Description
schedule	Schedule ID for M-Bus data collection

5.3 Wireless M-Bus

When equipped with a Wireless M-Bus transceiver (manufacturing option) the gateway can receive data from known Wireless M-Bus modules in mode S1, T1 (868MHz) and draft[1] N1 (169MHz).
The following parameters in `/config/metering/wmbus` are available:

Name	Values	Description
mode	S1, **T1**, N1	Wireless mode
long_preamble	true, **false**	S-mode radio preamble length
channel	**1-7**	N-mode channel (respectively N1a-N1g)
bypass_filter	true, **false**	Accept data from unknown modules
modules/*		List of Wireless M-Bus modules.

If OMS encryption is activated, the number of modules is limited to 64. Modules without encryption keys will be ignored.
Each module is configured as follow:

Name	Description
module/address	Wireless M-Bus address
module/label	Informative-only module’s name
module/key	Module’s encryption key

6 Active RFID tags

When equipped with an RFID receiver (manufacturing option) the gateway can collect data from active RFID tags.

[1] N1 mode is a 169MHz mode being added to the next revision of the standard EN 13757-4. It should not be considered final.

The receiver is compatible with active tags from ELA innovation[2]. The tags must be configured in 24-bit mode with 16-bit radio checksum.
The gateway receives the periodic transmission of all active RFID tags.
An optional CRC offset can be configured to filter out all tags not configured with the same CRC offset.
All IDs received with an RSSI above the configured threshold are ignored[3]. This threshold makes it possible to reduce the coverage area.
The following parameters in `/config/rfid` are available:

Name	Values	Description
rssi_threshold	0-**255**	RSSI level for filtering distant tags
crc	**0**	CRC optional offset
alarm/sources/id_flags	on, off, **delayed**	Alarm when a ID-only tag has a ID flag set
alarm/sources/data_codes		Handling of special DATA values (see below)
decimation	**1**	Decimation factor

Alarms

If the tag is an ID-only tag (the most significant bit of its ID is 0), the next 3 bits are processed as alarm flags and therefore not considered as being part of the ID.
If the tag is an ID+DATA tag (the most significant bit its ID is 1), the first 12 bits are used as the ID and the following 12 bits are considered to be data. These tags do not use flags to indicate alarm states. Instead a special data value is sent alternately with the normal value. In order to handle these special values, they can be listed in `/config/rfid/alarm/sources/data_codes/` as follow:

Name	Values	Description
code/label		Informative-only data code name
code/id_mask		Mask used to select a subset of tag IDs
code/id_value		Value used to select a subset of tag IDs
code/data_value		Special DATA value to be handled as an alarm
code/mode	on, **off**, delayed	Alarm mode
code/reset_count		Number of consecutive normal DATA values to consider the end of an alarm

For example to handle the low-battery alarm of T/RH tags with IDs 8xx and 9xx, we can add the following data code:

Name	Value
code/label	low-battery
code/id_mask	0xE00
code/id_value	0x800
code/data_value	0x7FF
code/mode	on

[2] http://www.ela.fr
[3] The RSSI range is 110-200 (close-far). If the threshold is set to a value smaller than 110, no data will be received. If the threshold is set to a value larger than 200, all received data will be processed.

The id_mask and id_value parameters will match any tag with an ID 0x8xx or 0x9xx and no other. When a value of 0x7FF is found for one of those tags, it will be handled as an alarm.

Decimation factor

In order to limit the amount of data uploaded to the remote server, a decimation factor can be configured. After each scheduled upload, for a given tag, the first data received is stored while the next [`/config/rfid/decimation` minus one] data received are not. In the uploaded data, a count value is added to keep track of the number of ignored data. This value should match (minus one) the decimation factor value except for the last one before the scheduled upload.
When the decimation factor is set to 0, only the first received data will be logged. When set to 1, all received data will be logged. When set to 2, one out of two received data will be logged.

RFID Listen

In order to facilitate the installation of new tags, the user interface contains a "RFID Listen" button which when clicked display a popup containing a list of tags along with date of the last received data and the associated RSSI level.
Note:
Data received from ID+DATA tags are logged as raw data. They are not converted to temperature/humidity/movement since the gateway is not aware of its type.

7 Modbus

Acting as a Modbus Master device, the gateway can read/write registers on Modbus RTU and TCP slave devices.
The gateway Modbus configuration consists of a list of datasets and a list of devices. A dataset is a list of registers for a given type of Modbus slave device. The list of devices associates a Modbus slave device with a dataset and a schedule.
In polling mode, the value of all variables will be continuously updated. These values can be monitored for changes or compared to threshold levels.
The current set of values will be logged when:

- The value of a monitored variable changes or cross a threshold level,
- The associated schedule occurs.

In snapshot mode (i.e. not in polling mode), the value of all variables will be updated and logged when and only when the associated schedule occurs.
Independently of the Modbus data collection process, it is possible to write to registers of a Modbus slave modules by using the Modbus command (see §0).

7.1 Configuration

The Modbus configuration in `/config/modbus` contains the following parameters:

Parameter	Values (default)	Description
tcp/timeout	**1000**	Modbus/TCP response timeout in ms
rtu/timeout	**1000**	Modbus/RTU Response timeout in ms
rtu/turnaround	**100**	Modbus/RTU Turnaround delay in ms
datasets/*		List of datasets
modules/*		List of modules

In addition to these parameters, the parameters `/config/system/ports/rs485/` must be set correctly (see §8.2). In particular the parameter `/config/system/ports/rs485/mode` must be set to “modbus”.

7.2 Modbus datasets

The configuration of a Modbus dataset (`/config/modbus/datasets/dataset`) consists of the following parameters:

Parameter	Description
id	Unique Modbus dataset identifier.
label	Dataset informative name.
vars/*	List of variables.
boundaries/*	List of register boundaries.
polling	Continuous polling (true or false)

Variables

Each variable is specified in `/config/modbus/datasets/dataset/vars/var` by the following parameters:

Parameter	Description
name	Variable name (informative only)
type	Variable type (S0, S1, S3, S4)
address	16-bit wide register address
size	Size in bits for discrete input and coil, in bytes for registers
format	See list below
flags	Comma-separated list of flags (see flag definition below).
threshold/low	Low threshold level (see details below).
threshold/high	High threshold level (see details below).
threshold/hysteresis	Hysteresis applied to both threshold levels.

Parameter “type”

The type of a variable is one of the four Modbus register types.

Type	Description	Read (multiple)	Write (single)	Write (multiple)
S1	Discrete input	0x02	-	-
S0	Coil	0x01	0x05	0x0F
S3	Input register	0x04	-	-
S4	Holding register	0x03	0x06	0x10

In the table below, the read/write function codes are given for information only. The Modbus requests are not part of the configuration but will be deduced from it. In particular multiple Read function codes will be used wherever it lowers the communication overhead.

Parameter "address"
This document always refers to Modbus register addresses (starting at 0) and never to Modbus register number (starting at 1).

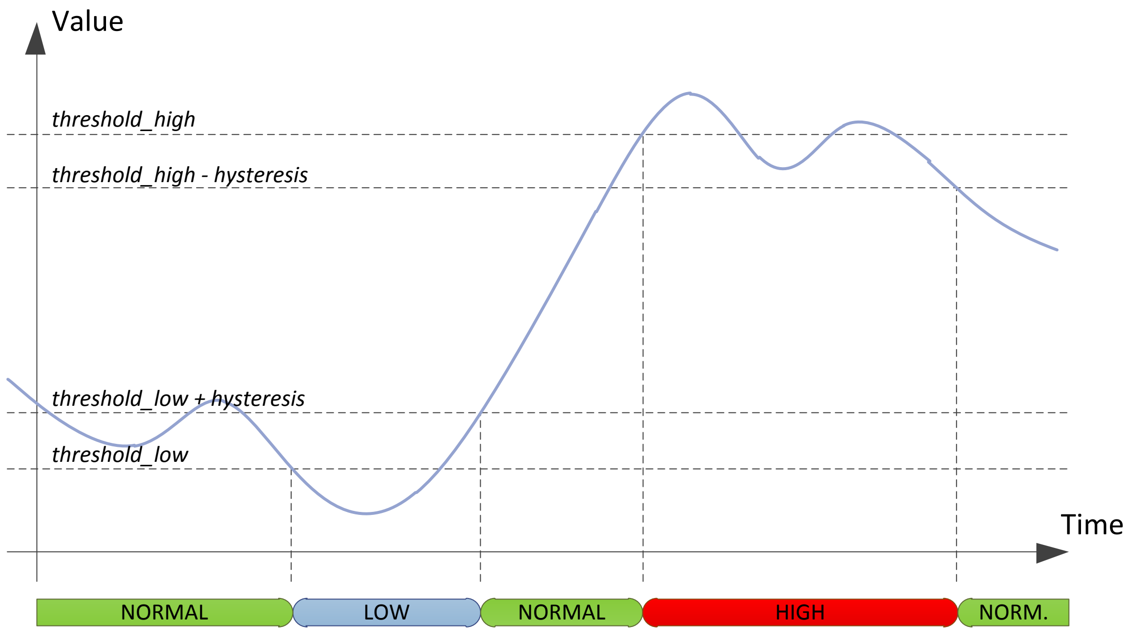
Parameter "format"

Format	Description	Coil	Register
raw	The data will be represented as a binary string for discrete inputs and coils and as a hexadecimal string for registers	✓	✓
boolean	False or true	✓	
integer	8, 16 or 32-bit integer		✓
float	16 or 32-bit floating point (IEEE 754)		✓
ascii	String of ASCII characters.		✓

Parameter "flags"

Flag	Description
cmd_only	The variable will not be read from the Modbus device, but can be written to.
little_endian	Interpret the two 16-bit registers of a 32-bit value in little-endian order.
no_opt	A dedicated Modbus query will be used to read this variable.
signed	The variable contained a signed value.
is_status	Indicates that the variable contain a status information.
is_alarm	Any change to the variable status will trigger an alarm.

Parameter "alarm"
For float and integer variables, two threshold levels can be defined (alarm/low and alarm/high) along with a hysteresis value. Each time the variable is updated, its value is checked against this level to determine an associated status (low, normal, high) as shown below:



Flags is_status and is_alarm

When at least one threshold level is set for an interger or float variable, an associated status is maintained (as explained above).

When the is_status flag is set, the variable is considered itself to contain a status.

In both cases, in polling mode, for any change to the status of a variable, the complete dataset will be logged.

In both cases, in both modes, for any change to the status of a variable with the is_alarm flag set, an alarm will be generated and a connection requested.

The flag is_alarm has no effect unless the variable has its is_status flag set or has at least one threshold level defined.

Additional data in polling mode:

In polling mode, additional data will be maintained for integer and float values: The min/max/average values and the number of samples since the last datalogging.

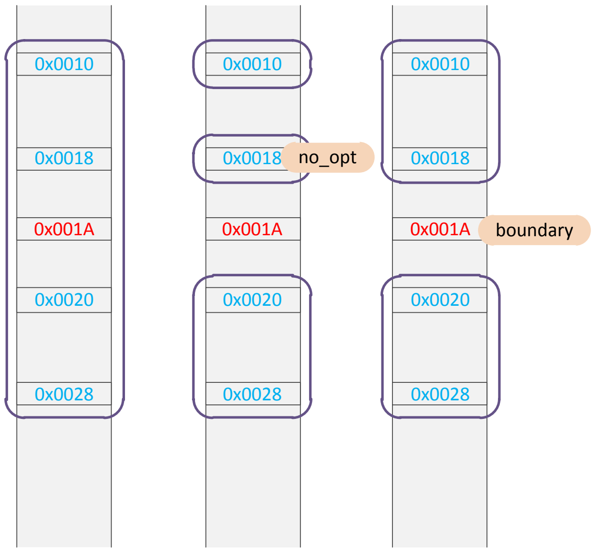
Boundaries (Not in v2.x)

By default the gateway will optimize the Modbus requests sent to a given device, reading multiple registers at once whenever possible.

In some cases, this may not be possible as some of the additional registers read by this optimized queries may not be available (the device would send back an exception) or may have side effects (for example reading a register may trigger an action such as clearing its content).

For these cases, it is possible to configure address boundaries. The optimization process will not cross these boundaries.

For example, let's consider a dataset with 4 "holding register" variables with the addresses 0x0010, 0x0018, 0x0020 and 0x0028. Normally the gateway would read all 4 registers with one read query, reading all registers from 0x0010 to 0x0028 (first row in the drawing below). Now let's imagine that the register at the address 0x001A must not be read. One solution would be to add the 'no_opt' flag to at least one of the two middle variables (0x0018, 0x0020) (second row). But this would prevent the gateway from optimizing the requests around. We can define a boundary address at 0x001A (third row).



Each boundary address is specified in `/config/modbus/datasets/dataset/boundaries/boundary` by the following parameters:

Parameter	Description
type	Type (S0, S1, S3, S4)
address	16-bit wide register address

7.3 Slave modules

A module is an instance of a dataset for a given Modbus address. The configuration of a Modbus module (`/config/modbus/modules/module`) consists of the following parameters:

Parameter	Description
label	Informative-only name
dataset	Modbus dataset identifier
address	Modbus address (1-247)
ip	IP address (empty for RTU devices)
schedule	Schedule identifier

Note: Modbus/TCP devices must be configured to listen on the default Modbus TCP port (502).

7.4 Variable address

The modbus command uses addresses formatted as explained below:

Modbus/RTU

    <modbus_address>/<register_type>@<register_address>

    Example:

        Input register at address 0x0056 on the Modbus device 45

        => 45/S3@0x0056

Modbus/TCP

`<device_ip>:<modbus_address>/<register_type>@<register_address>`

Example:

Input register at address 0x0F0C on the Modbus device 223 on 192.168.0.17

=> 192.168.0.17:223/S3@0x0F56

Note: Both the modbus_address and the register_address can be either in decimal or hexadecimal form. The later must be prefixed with "0x".

8 System configuration

8.1 Local access control

The local HTTP and FTP services are protected by login/password. All tentative to use these services are logged.

Three levels of credential are available: admin, install, data.

The administrator (admin) is given read/write access to all configuration parameters, read access to all status information and can trigger all actions.

The installer (install) is given read/write access to configuration parameters related to the end modules including the (de)activation of the Wavenis bridge mode, read access to all status information and can trigger all actions.

The data user (data) is only given read access to the gateway status.

	Gateway configuration	Gateway status	End modules configuration	Actions
admin	R/W	R	R/W	Yes
installer		R	R/W	Yes
data		R		No

The passwords associated with these users are configured in `/system/password`. They only can be changed by a configuration file either from the remote server or locally by the administrator.

It is strongly advised to change these default passwords prior to deployment.

When a configuration file is uploaded to the local FTP server by "installer", it will be rejected if it contains parameters not related to the WAN configuration.

The Wavenis TCP bridge port is not password protected. But it can be activated/deactivated by the "installer".

8.2 Ports configuration

The following parameters in `/config/system/ports` are used to configure the mode of each port:

Name	Values	Description
rs232/mode	**off**, mbus	RS232 mode
rs485/*	**off**	RS485 port parameters (see below).
input_1/mode	**d_input**, pulse	Digital input mode
input_2/mode	**d_input**, pulse	Digital input mode
input_3/mode	**d_input**, pulse	Digital input mode

The RS485 port parameters, in `/config/system/ports/rs485/`, are:

Parameter	Values (default in bold)
mode	**off**, modbus
baudrate	4800, 9600, **19200**, 38400, 57600, 115200
data	8
parity	odd, **even**, none
stop_bit	**1**, 2

Note: The Modbus specification requires that if no parity bit is used, 2 stop bits must be used.

9 Communication between the gateway and the remote server

The gateway can communicate with a remote server thru a FTP server and/or a Web Service server. Each time its configuration is modified, the gateway can either upload it on a FTP server or send it to a Web Service server.
Similarly alarms and collected data can be uploaded to a FTP server or sent to a Web Service server.
Moreover when using FTP, the gateway can notify a Web Service server of any FTP download/upload.
The server can initiate actions on the gateway by placing command files in an INBOX directory on the FTP server or by providing the commands when the gateway queries the INBOX web service.
Commands can also be sent to the gateway by SMS.

9.1 Modes of connection

The connection to the remote server can be established thru Ethernet or a cellular network (GPRS or 3G depending on the hardware configuration). The exchanges between the gateway and the remote server are always initiated by the gateway but different methods are available for the remote server to trigger an exchange (see paragraph 9.6).

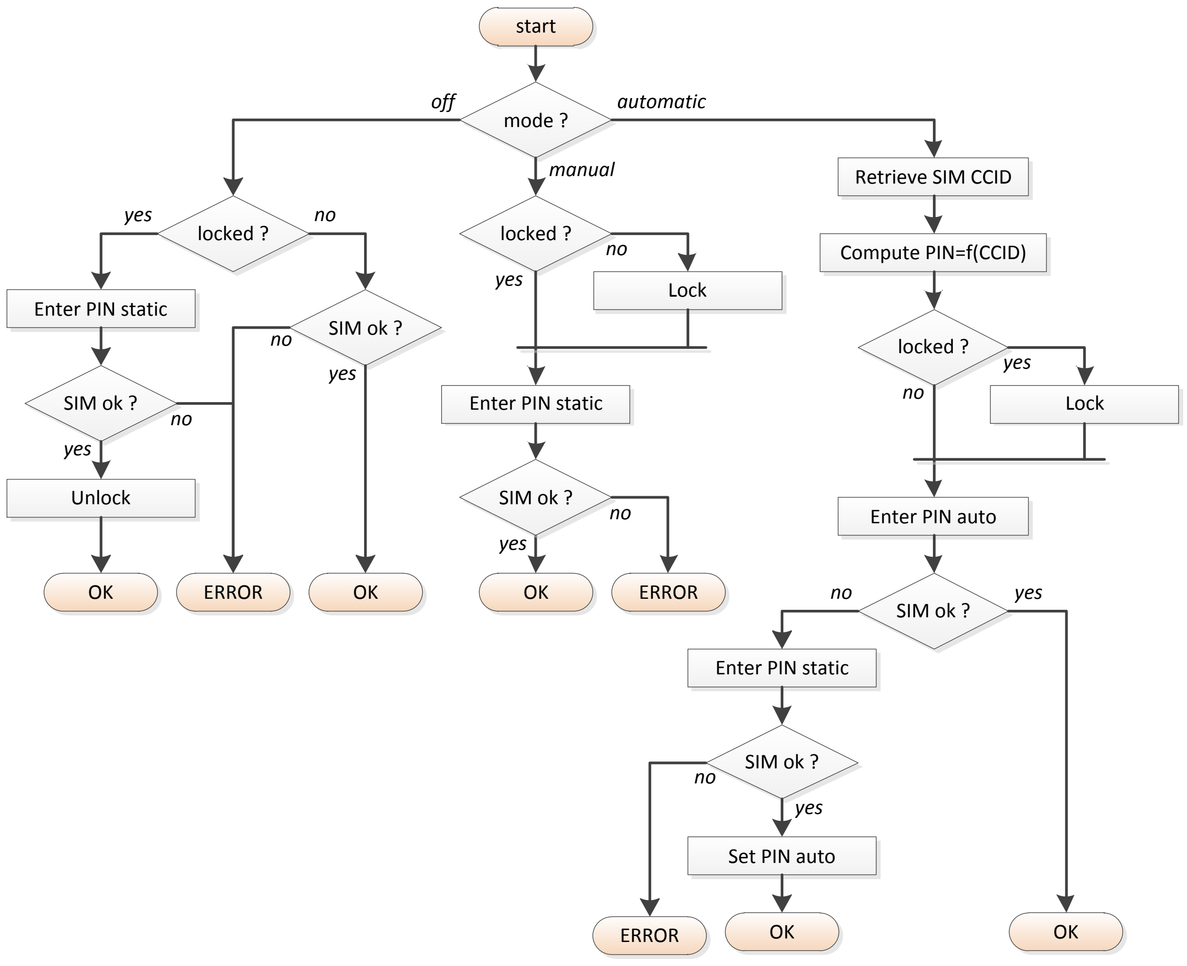
The gateway can be configured to use the modem in one of the following 4 modes:

- **On Demand**: In this mode, the PPP link will be created when the gateway needs to communicate with the server. The link will be shut down after the completion of the communication with the remote server, after a configurable delay. This delay is designed to avoid successive connection cycles, for example in case of successive alarms.
- **Always On**: In this mode, the PPP link will be maintained all the time independently. In this mode, a keepalive mechanism can be activated to make sure the link is functional.
- **Always Off**: In this mode, the PPP link is never created. All communications with the server go thru the Ethernet interface. The modem is connected to the cellular network, ready to receive incoming calls and/or SMSs.
- **Off**: In this mode, the modem is powered off.

The gateway can be configured to use a SIM card:

- without PIN code: `/com/modem/pin/mode=off`
- with PIN code: `/com/modem/pin/mode=manual` & `/com/modem/pin/code=<CODE>`
- with automatic PIN code: `/com/modem/pin/mode=automatic`

In the later mode, the gateway will generate a PIN code and assign it to the SIM. The initial PIN code of the SIM card should be given by `/com/modem/pin/code` (by default 0000).



9.2 FTP

The gateway uses the following files on the FTP server:

Name	Description
CONFIG/<uid>.xml	Current gateway configuration. This file is uploaded by the gateway after each modification of its configuration. Modifying this file has no effect on the gateway. The gateway will only overwrite it the next time its configuration is modified (See INBOX below).
DATA/<uid>-<timestamp>.<format>.gz	Data file uploaded by the gateway.
ALARM/<uid>-<timestamp>.xml.gz	Alarm file uploaded by the gateway.
SUPERVISION/<uid>-<timestamp>.xml.gz	Supervision files uploaded by the gateway (status and scan results).
SUPERVISION/<uid>-<timestamp>.log.gz	Log files uploaded by the gateway upon request.
INBOX/<uid>/*.xml	The gateway monitors this directory. Any file placed in this directory will be downloaded and processed by the gateway.
BIN/<firmware>	This directory contains the available firmware.

In the above table, <uid>, <timestamp> and <format> should be replaced respectively by the gateway unique identifier, the upload timestamp and the selected format (CSV or XML).
The timestamp format is « YYYYMMDD-hhmmss » so that an alphabetical sort of the directory gives the chronological order.
The files with the .gz extension are compressed.
The gateway always uploads files following a 2-step process:

- The file is uploaded with an additional .tmp extension.
- The file is renamed by stripping the .tmp extension.

This process allows the remote server to easily distinguish files being uploaded from files completely uploaded.
If the "/com/ftp/secured" parameter is set to "true", explicit FTPS connection will be used. To use implicit FTPS connection (for older FTP server), it is necessary to prefix the server address with "ftps://".

9.3 Web Service

The gateway can interact with a REST Web Service server.
Three types of services are supported:

- Notification: The gateway notifies the WS of all FTP actions.
- Upload: The gateway upload data and alarms.
- Inbox: The gateway asks the WS if any actions are pending.

Below, <ws_address> is a placeholder for the WS address (parameter `/com/ws/address`). The address can include a TCP port and a path.

Notification

When enabled, the gateway will notify the WS when it downloads or uploads a file from/to the FTP server.
After uploading a file, the gateway will notify the WS by sending an HTTP POST request to the following URL: http://<ws_address>/notify?uid=<uid>&put=<file>.
After downloading a file, the gateway will notify the WS by sending an HTTP POST request to the following URL: http://<ws_address>/notify?uid=<uid>&get=<file>.
Notifications can be selectively activated with the parameter `/com/ftp/ws_notification`.

Upload

The gateway can be configured to selectively upload its current configuration, alarms and data.

Content	Activation	WS URL
Configuration	/upload/config/method=ws	http://<ws_address>/config?uid=<uid>
Alarm	/upload/alarm/method=ws	http://<ws_address>/alarm?uid=<uid>
Supervision	/upload/supervision/method=ws	http://<ws_address>/supervision?uid=<uid>
Data	/upload/data/method=ws	http://<ws_address>/data?uid=<uid>

In all cases, the body of the HTTP request will contain the XML data and will be considered uploaded only after receiving a success HTTP status code (Type 2xx).

Inbox

Each time the gateway connects to the WS server, it will ask if there are pending actions.
To do this, the gateway will send an HTTP POST request to the following url:

http://<ws_address>/inbox?uid=<uid>

If the WS sends back XML data, it will be processed by the gateway. The XML data must be a valid configuration or command file. Both file formats are described later in this document. The available commands are described in the paragraph 9.5.

9.4 SMS

When the gateway receives an SMS, it checks the SMS callerID whitelist. If authorized, the SMS content is processed.
The commands and associated formats are described in the next paragraph.

9.5 Commands

Command	Description	Feedback
reboot	Restart the gateway.	None
factory	Restore the gateway to its factory settings.	None
update	Update the firmware of the gateway.	SW alarm
scan	Request to retrieve from some Wavenis modules instantaneous values, RSSI levels, battery levels and/or RTC values.	Supervision data (except for scan data)
timesync	Request to update the RTC value of some Wavenis modules.	Supervision data
wavenis	Wavenis specific command (see below)	Alarm
modbus	Modbus specific command (see below)	Alarm
status	Request to send back the gateway status.	Supervision data
log	Request to upload log data.	Log data
d_output	Change the state of the digital output	Alarm
config	Modification of the gateway configuration (SMS only).	Configuration upload
connect	Triggers a connection to the server (SMS only)	Implicit (connection)

Commands are not acknowledged when received. All commands are logged and an invalid command triggers an alarm which is uploaded to the remote server.
All commands accept two optional parameters “uid” and “cid”:

- uid: Gateway unique identifier
- cid: Command identifier

A command will be rejected if the uid parameter is included and does not match the gateway uid.
The cid can be freely selected by the command emitter. It will be included with any associated upload.

Wavenis commands: Scan, TimeSync

A list of Wavenis modules can be specified for the commands « Scan » et « TimeSync ». Otherwise, the command is applied to all known modules. Commands can only be applied to known modules (all unknown addresses are ignored).
The requested data (except for the immediate index data) will be uploaded to the remote server as supervision data using the configured upload method (/upload/supervision/method). The immediate index data (scan data) will be uploaded to the remote server as wavenis data using the configured upload method (/upload/data/method).

Example: Request to update the RTC of two Wavenis modules:

```
XML:
<cmd cid='C_1234'>
   <timesync>
      <address>011A0A30AAA0</address>
      <address>011A0A30AAA1</address>
   </timesync>
</cmd>
SMS:
cmd=timesync
cid=C_1234
address=011A0A30AAA0
address=011A0A30AAA1
```

The « Scan » command must specify the types of information requested, among:

- data : Instantaneous data
- rssi : RSSI level
- life-counter : battery life-counter
- rtc : RTC value
- tic : profile configuration

For « rssi », « life-counter » and « rtc », the gateway will also retrieve the information from the repeater on the modules path.
For « data », a command alarm with error="none" will be sent to indicate to the server that the command has been completed.
Example: Request of the RTC value and battery life counter for two Wavenis modules

```
XML:
<cmd cid='C_1235'>
   <scan mode='rtc life-counter'>
      <address>011A0A30AAA0</address>
      <address>011A0A30AAA1</address>
   </scan>
</cmd>
SMS:
cmd=scan
cid=C_1235
mode=rtc, life-counter
address=011A0A30AAA0
address=011A0A30AAA1
```

Example: Request of instantaneous data from all known Wavenis modules

```
XML:
<cmd cid='C_1236'>
   <scan mode='data'/>
</cmd>
SMS:
cmd=scan
cid=C_1236
mode=data
```

Wavenis specific commands

Wavenis specific commands can be sent to a known Wavenis module using the "wavenis" command. The gateway will use the repeaters configured for the module.

The list of supported sub-commands is:

Sub-command	Description
moduflow-open	Request the opening of the valve attached to a moduflow.
moduflow-close	Request the closing of the valve attached to a moduflow.
moduflow-state	Retrieve the open/close state of the valve attached to a moduflow.
raw	Send a raw Wavenis command.

The 3 moduflow commands can be sent to any known wavenis module. The gateway will not check that the module supports the command but will report any error.

```
XML:
<cmd cid='C_1239'>
   <wavenis subcmd='moduflow-open'>
      <address>011A0A30AAA0</address>
   </wavenis>
</cmd>
SMS:
cmd=wavenis
cid=C_1239
subcmd=moduflow-open
address=011A0A30AAA0
```

The result of the command will be reported to the server as an alarm wavenis_cmd.
The result code contained in the alarm for the moduflow-open and moduflow-close commands can be: ok (the command was accepted by the module), error (the command was rejected by the module) and unsupported (the command is not supported).
The result code contained in the alarm for the moduflow-state command can be: open (the valve is opened), close (the valve is closed) and unsupported (the command is not supported).
The raw sub-command takes an additional parameter data that must be a hexadecimal string starting with a Wavenis applicative command. It will be sent in a Wavenis REQ_SEND_FRAME.
For example to read the 'applicative status' (0x20) byte from the known module (011A0A30AAA0), you can use the Wavenis applicative command 0x10 (Read parameter):

```
XML:
<cmd cid='C_1240'>
   <wavenis subcmd='raw' data='10012001'>
      <address>011A0A30AAA0</address>
   </wavenis>
</cmd>
SMS:
cmd=wavenis
cid=C_1240
subcmd=raw
data=10012001
address=011A0A30AAA0
```

The result code contained in the alarm for the raw command can be: ok (the command was accepted by the module), error (the command was rejected by the module).

If the command was accepted by the module, its response is added to the alarm:

```
<wavenis_cmd_alarm>
      <date>2011-05-27T20:00:00</date>
      <cid>C_1240</cid>
      <source>ftp</source>
      <subcmd>raw</subcmd>
      <address>011A0A30AAA0</address>
      <result>ok</result>
      <request>10012001</request>
      <response>9001200106</response>
</wavenis_cmd_alarm>
```

Modbus specific commands

Modbus specific commands can be sent to a known Modbus module using the “modbus” command. The list of supported sub-commands is:

Sub-command	Description
write	Write a value to a Modbus device

Variable addresses must be formatted as explained in §7.4.

```
XML:
<cmd cid='C_1239'>
   <modbus subcmd='write' data='27'>
      <address>192.168.0.17:223/S3@0x0F56</address>
   </modbus>
</cmd>
SMS:
cmd=modbus
cid=C_1239
subcmd=write
data=27
address=192.168.0.17:223/S3@0x0F56
```

The result of the command will be reported to the server as an alarm modbus_cmd.
The result code contained in the alarm can be: ok, no_response or error (if an exception function code was sent back by the module). In the latter case, the Modbus exception code will also be present in the alarm.

Status command

Example: Request to send back the gateway status

```
XML:
<cmd cid='C_1237'>
   <status/>
</cmd>
SMS:
cmd=status
cid=C_1238
```

The following information is sent back to the requester:

XML name	SMS name	Description
-	uid	Unique gateway identifier.
/app/version	version	Software version.
/app/kernel	kernel	Linux kernel version.
/system/power	power	External power supply availability (boolean)
/system/defaults	defaults	Comma-separated of default codes
/system/uptime	uptime	Time elapsed since the last boot
/com/modem/model	m_model	Modem model name
/com/modem/firmware	m_version	Modem firmware version
/com/modem/imei	imei	International Mobile Equipment Identity
/com/modem/msisdn	msisdn	Mobile Subscriber ISDN Number (if available)
/com/modem/rssi	rssi	Received signal strength
/com/modem/csq	csq	Signal quality (CSQ, BER)
/com/modem/ip	m_ip	IP address of the gateway on the PPP modem interface (or last assigned address).
/com/ethernet/ip	e_ip	IP address of the gateway on the Ethernet interface
/com/upload/last	u_last	Date of the last successful (periodic or triggered) connection to the remote server.
/com/upload/next	u_next	Date of the next periodic connection to the remote server.
/wavenis/address	w_addr	Wavenis address of the gateway
/wavenis/last	w_last	Time of the last Wavenis successful communication
/wavenis/modules/count	w_count	Number of Wavenis modules
/metering/mbus/last	mb_last	Time of the last mBus successful communication
/metering/mbus/last_scan	mb_lastscan	Time of the last mBus scan
/metering/mbus/modules/count	mb_count	Number of mBus modules
/metering/wmbus/radio/type	-	Wireless mBus radio type
/metering/wmbus/radio/firmware	-	Wireless mBus radio firmware revision
/metering/wmbus/radio/hardware	-	Wireless mBus radio hardware revision
/metering/wmbus/radio/serial	-	Wireless mBus radio serial number
/metering/wmbus/last	wmb_last	Time of the last mBus successful communication
/metering/wmbus/modules/count	wmb_count	Number of mBus modules
/rfid/modules/count	rfid_count	Number of tags in the zone
/rfid/tags	-	List of RFID tags
/modbus/next	-	Date of the next Modbus query
/modbus/modules/count	modbus_count	Number of Modbus modules

When the status command was sent by SMS, the status is sent back in multiple SMS with one variable per line (name=value).
When the status command comes from the FTP/WS inbox, the XML file is uploaded as supervision data.

Other commands

Example: Firmware update command

```
XML:
<cmd cid='C_1238'>
   <update>
      <firmware>wrf_wavenis_v101.bin</firmware>
      <checksum>c1fb7d81f3d53a8b7bf94098115249d3</checksum>
   </update>
</cmd>
SMS:
cmd=update
cid=C_1238
firmware=wrf_wavenis_v101.bin
checksum=c1fb7d81f3d53a8b7bf94098115249d3
```

The firmware file is expected to be available in the BIN directory on the FTP server (see paragraph 0).
The checksum is the file md5 checksum.
Example: Opening the gateway dry contact (digital output)

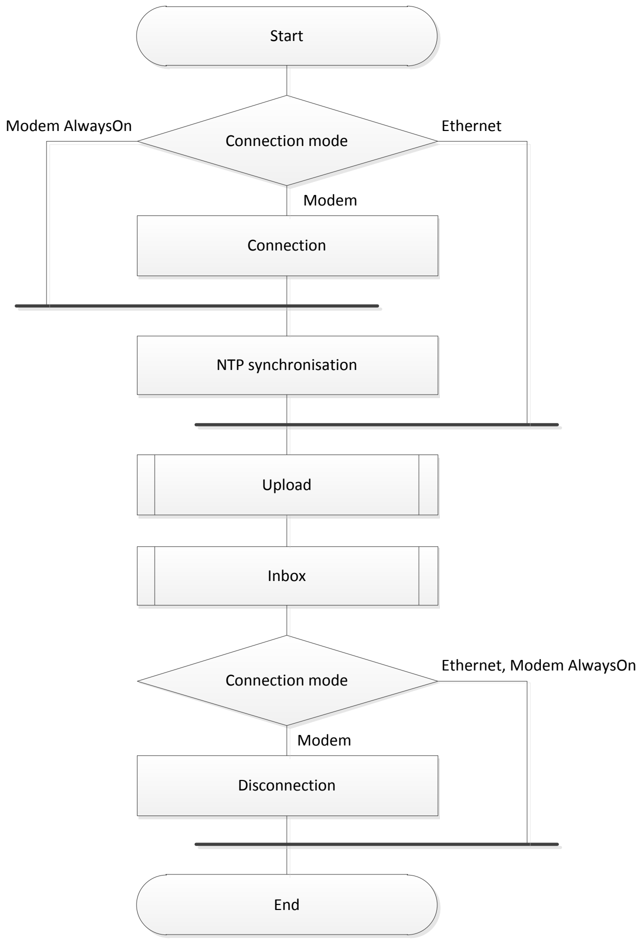
```
XML:
<cmd cid='C_1239'>
   <d_output subcmd='open'/>
</cmd>
SMS:
cmd=d_output
cid=C_1239
subcmd=open
```

9.6 Connection to the remote server

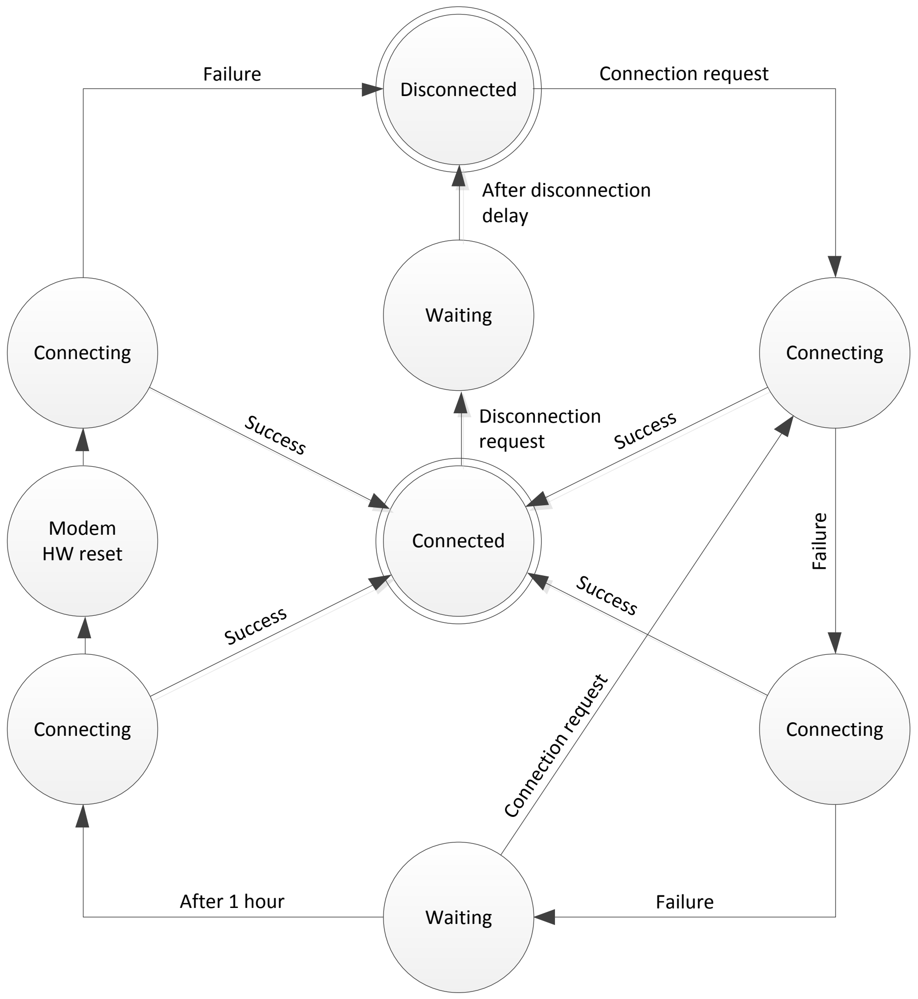
A connection to the remote server can be initiated by any of the following events:

- Scheduled data upload,
- Reception of an alarm from a Wavenis module,
- Configuration change,
- SMS request of connection,
- Incoming call from a whitelisted caller ID,
- Reception of any data on the TCP keepalive link,
- Pushing the connection button on the gateway.

Independently of the triggering event, the following process is executed:



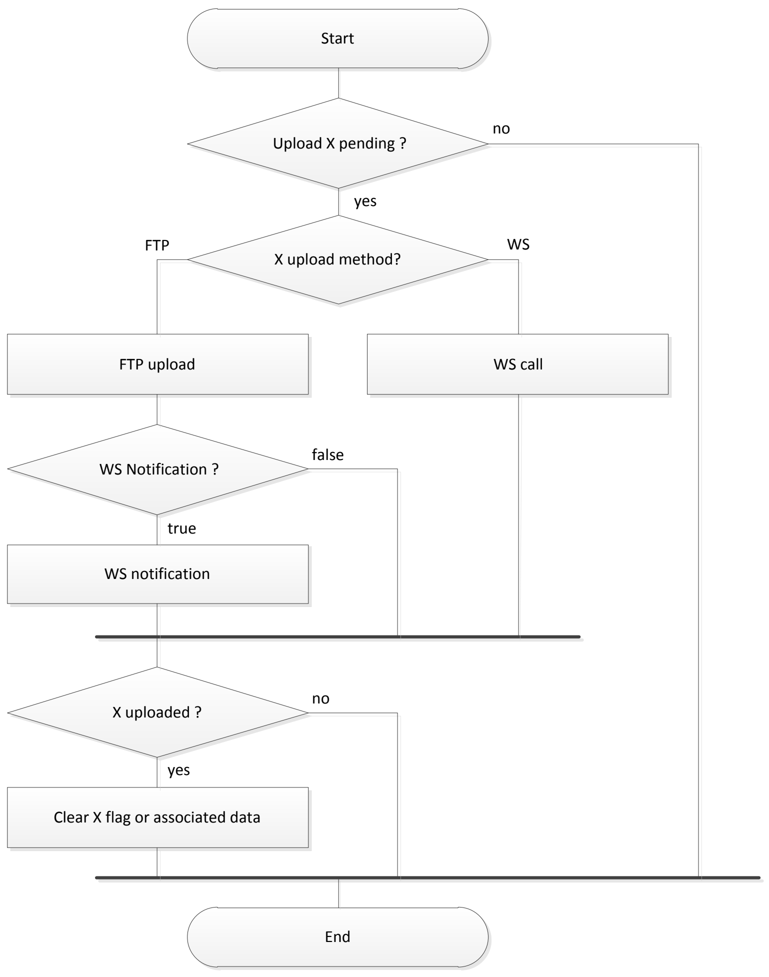
In case of failure of the GPRS data connection, a new connection will be tried one hour later unless a new connection was initiated in the meantime (by explicit request, or triggered by an alarm or by a periodic upload). This is illustrated below including the delayed disconnection.



If 10 iterations of the loop "Connecting -> Connecting -> Waiting" (the 3 states on the bottom right) is detected, the transition to the next state is forced without waiting 1 hour.

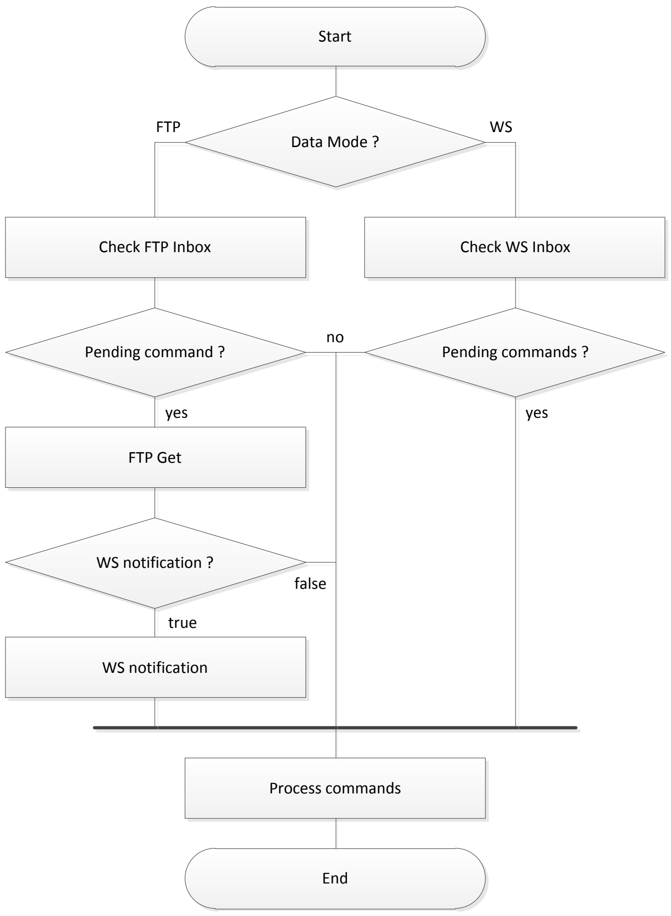
9.7 Upload process

Configuration, Alarm, Supervision and Data are uploaded independently as described in the below diagram. In this diagram, X stands for the type of upload (Configuration, Alarm, Supervision and Data). After the completion of a configuration upload, the associated flag is cleared. After the completion of an alarm/supervision/data, the associated data is cleared.



9.8 Inbox

The gateway checks for pending actions as follow:



9.9 Keepalive

The keepalive is activated when:

- The connection type is Ethernet
- The connection type is modem and the mode AlwaysOn is selected.

The gateway monitors the connection with one of the two following methods:

ICMP method

This method is used when the parameter `/com/keepalive/method` is set to « ICMP ». The gateway will periodically send ICMP packets « echo request ». The connection is considered as valid if at least one « echo reply » ICMP answer is received during the configured timeout (`/com/keepalive/timeout`).

Méthode TCP

This method is used when the parameter `/com/keepalive/method` is set to « TCP ».

A TCP connection is created by the gateway. The remote address and TCP port are configured with the parameters `/com/keepalive/ip` et `/com/keepalive/port`. Once the TCP connection established, the gateway sends periodically empty TCP packets. The IP connection is considered valid as long as the TCP connection is valid.

In this mode, any data received on this TCP connection will be interpreted as a connection request (see paragraph 9.6). The received data will not be processed at all.

9.10 Request button

By default pressing the “Request” button will trigger a connection to the remote server and upload, in addition to pending data, a status. Both can be selectively deactivated with the configuration parameters /com/request/upload and /com/request/include_status.
A status SMS will also be sent to the specified recipient (/com/request/sms_status_recipient), if any.

10 Scheduler

The scheduler is in charge of all periodic tasks.
The scheduler configuration consists of a list of schedules.
Each one of these schedules has a unique ID which is used to link a task to a specific schedule.
Schedules can be used independently to trigger data collection and upload of collected data.

Name	Description
/scheduler/schedules/schedule	Each schedule configuration (described below) will be stored under this element.

Each schedule is configured as follow:

Nom	Description
schedule/id	Schedule unique identifier
schedule/label	Informative-only schedule name
schedule/type	Type: day, week, month, year, follow
schedule/parent	Reference to the parent schedule for a schedule of type “follow”.
schedule/start/time	Time of first occurrence in a given period (not used for yearly schedules)
schedule/start/datetime	Date of first occurrence in a given period (only used for yearly schedules).
schedule/start/dayofweek	Day number of the first occurrence in a week (1=Monday, 7=Sunday) (only used for weekly schedules).
schedule/start/dayofmonth	Day number of the first occurrence in a month (only used for monthly schedules).
schedule/interval	Interval between occurrences (in seconds)
schedule/count	Number of occurrences

Daily schedule
Every day, the first occurrence T_0 is given by `schedule/start/time`.
The next occurrences are at T_i :

$$T_i = T_0 + i \times \Delta t \qquad \begin{cases} i < count \\ \forall i \; jour(T_i) = jour(T_0) \end{cases}$$

Weekly schedule
Every week, the first occurrence T_0 is given by the week day number `schedule/start/dayofweek` at the time given by `schedule/start/time`.

The next occurrences are at T_i :

$$T_i = T_0 + i \times \Delta t \qquad \begin{cases} i < count \\ \forall i \; semaine(T_i) = semaine(T_0) \end{cases}$$

Monthly schedule

Every month, the first occurrence T_0 is given by the day of the month `schedule/start/dayofmonth` at the time given by `schedule/start/time`.

The next occurrences are at T_i :

$$T_i = T_0 + i \times \Delta t \qquad \begin{cases} i < count \\ \forall i \; mois(T_i) = mois(T_0) \end{cases}$$

Yearly schedule

Every year, the first occurrence T_0 is given by `schedule/start/datetime`.

The next occurrences are at T_i :

$$T_i = T_0 + i \times \Delta t \qquad \begin{cases} i < count \\ \forall i \; année(T_i) = année(T_0) \end{cases}$$

Follower schedule

A schedule of type “follow” will occurs after the completion of each occurrence of the referenced schedule. The referenced schedule can’t have the type “follow”.

This type of schedule allows triggering a data upload after the completion of a scheduled data collection.

Example:

Let say you want to collect the datalog of all Wavenis modules once a day at midnight and upload the data just after. You can configure a daily schedule (#1) for the data collection and a schedule (#2) following #1 (i.e. the first schedule completion) for the data upload.

```
<schedules>
   <schedule>
      <id>1</id>
      <label>Data collect</label>
      <type>day</type>
      <start>
         <time>00:00:00</time>
      </start>
   </schedule>
   <schedule>
      <id>2</id>
      <label>Data upload</label>
      <type>follow</type>
      <parent>1</parent>
   </schedule>
</schedules>
```

11 Alarm engine

The alarm engine generates alarms based on internal events.

Each alarm source can be individually enabled and will be uploaded immediately to the remote server (on) or at the next connection (delayed).

Name	Values	Description
sources/power	on, **off**, delayed	Main power supply state change
sources/modem_ip	on, **off**, delayed	IP changed
sources/msisdn	on, **off**, delayed	MSISDN changed
sources/sw_version	**on**, off, delayed	SW version changed
sources/defaults/ignored	***<empty>***	Comma-separated list of default codes that will not trigger an alarm
sources/defaults/delayed	***<empty>***	Comma-separated list of defaults that will not trigger a connection
sources/d_inputs/*		Digital input sources (see below).
sources/d_output		Digital output sources (see below).

The default codes are:

Code	Description
D_MODEM	Modem default
D_MODEM_PUK	Carte SIM bloquée
D_ETHERNET	Ethernet interface default
D_WAVENIS	Wavenis radio default
D_RFID	RFID receiver default
D_INTERNAL_BAT	Internal battery default

When the “modem_ip” source is enabled, the alarm engine will send an alarm containing the gateway IP whenever it changes.
When the “sw_version” source is enabled, the alarm engine will send an alarm containing the gateway software version after an update.
Multiple alarms can be configured for the digital inputs. An alarm for a digital input can be configured as follow:

Name	Values	Description
d_input/index		Index of the digital input.
d_input/label		Informative-only alarm name.
d_input/mode	on, off, delayed	
d_input/type	none, raising, falling, both	Type of detection (raising/falling edges).

An alarm can be configured for the digital output as:

Name	Values	Description
d_output/label		Informative-only alarm name.
d_output/mode	on, off, delayed	
d_output/type	none, raising, falling, both	Type of detection (raising/falling edges).

The raising edge corresponds to the closing of the gateway dry contact.

Alarms are uploaded to the remote server in XML as specified by the alarm XML schema (see Appendix C).

12 Log

The gateway logs the main events in log files. The size of the log file is limited to 200kB. Upon request the previous and current log files are concatenated and uploaded to the remote server as supervision data.
The log file is a text file with one log entry per line. Each line is formatted as follow:

[TIMESTAMP][LEVEL][SOURCE] EVENT

The timestamp is a unix timestamp (time since EPOC) in seconds followed by a point and the microseconds.
The following events are logged:

Event	Format
Schedule event	Schedule X occurred
Default events	Default X detected
	Default X cleared
PPP events	PPP connecting
	PPP connected IP=X
	PPP connection failure
	PPP disconnect
FTP events	FTP connecting
	FTP connected
	FTP connection failure
	FTP get X
	FTP put X
	FTP disconnect
SMS event	SMS received from X
Command events	Processing command X
Clock events	Clock synchronization delta=X
	NTP connection failure
Internal events	Reboot
	Reboot modem

The list is not exhaustive and each line can contain additional information after the one specified above. For example, when a schedule occurs the log may contain “Schedule X occurred, next is Y at Z”.

The level of details logged can be configured with the parameter `/system/log/level`. Its value must be between 8 (no log) to 1 (verbose):

Level	Name	Description
8	None	Nothing is logged
7	Critical	Less verbose log level (critical info only).
6	Error	
5	**App**	**Default log level**
4	Warning	
3	Notice	
2	Info	
1	Debug	Most verbose log level.

The log level parameter can also be used to set a per-source level. The format is then as follow:

default_level,source:level,source:level,…

For example to use the default level 5 for all sources but 1 for the source “Coronis”:

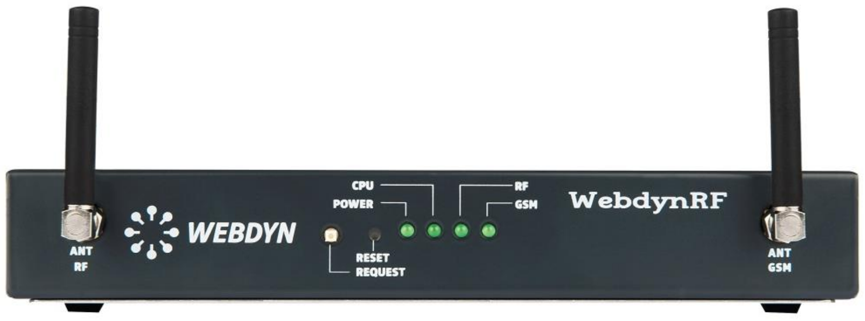
```
/system/log/level=”2,Coronis:1”
```

13 Clock synchronization

The gateway synchronizes it system clock using the NTP protocol.
It keeps time in UTC (Universal Time, also known as GMT) and calculates a local time based on the configured timezone. It handles the daylight saving time (DST).
When the connection uses the modem, the synchronization is done at the beginning of each connection to the server but not more than once a day.
An alarm is triggered when the difference between both clocks is larger than a configurable parameter.
When the connection uses Ethernet, a NTP client is activated on the gateway. This client will adjust the system clock speed in order to keep it synchronized with the NTP server clock.

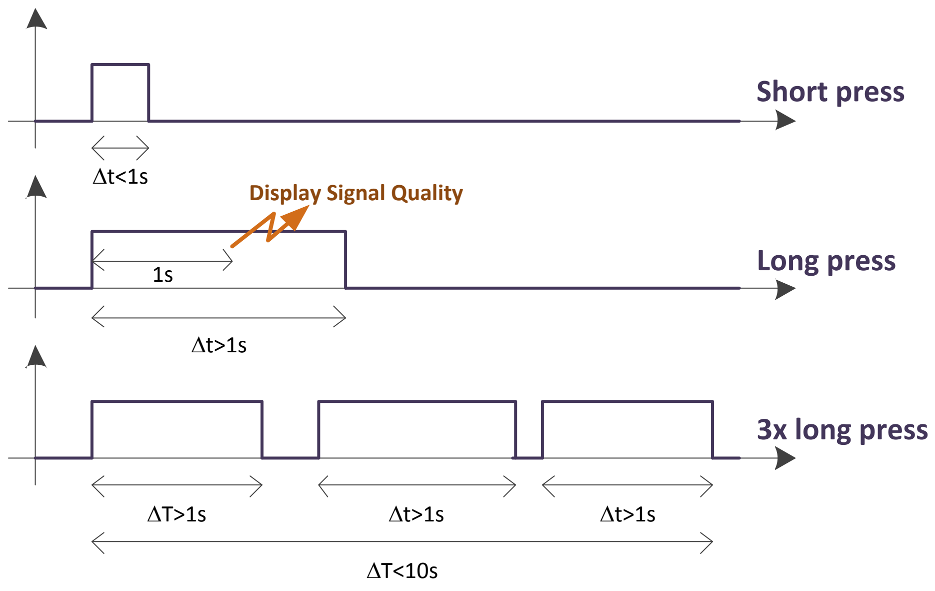
14 User interface

14.1 IHM



The gateway is equipped with 2 buttons: Request, Reset.

- The reset button is a pin-hole button. Pressing it causes a hardware reset of the gateway.
- A short press on the request button initiates a connection to the remote server and forces the upload of the configuration and status files.
- A long press on the request button will cause the Modem LED to display the GPRS/3G signal quality. The signal quality will be displayed after 1 second.
- 3 successive long presses on the request button will trigger a restore the gateway factory settings. The button must be pressed for longer than 1 second, 3 times in less than 10 seconds.



The gateway is equipped with 4 LEDs: CPU, RF, GSM and Power.

- The power LED is on when the external power supply is present (independently of the state of the internal battery).
- The CPU LED blinks slowly to show the CPU activity and is off when the gateway is in low power mode.
- The GSM LED blinks when data is received or transmitted thru the modem and is on for 1 second when an SMS is received.
- The RF LED blinks when data is received or transmitted thru the Wavenis radio.

14.2 Web interface

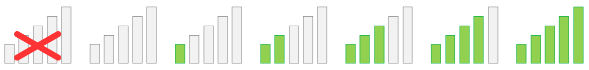
The web interface is available on TCP port 80 of the gateway. It is compatible with the latest version of all major browsers (older versions may work but are not supported; Internet Explorer 7 released in 2007 is not supported).
It can be used to display the status of the gateway and to edit its configuration.

Overview panel

The following information is displayed in the Overview panel:

- Gateway: UID, Name, Firmware, Kernel
- Modem: Model, Firmware, IMEI, MSISDN, RSSI, CSQ, IP
- Ethernet: IP
- System: External power supply, Defaults
- Upload: Last upload, Next upload
- M-Bus: Last, Last scab and module count
- Depending on the radio:
    - Wavenis: Address, Last, module count
    - Wireless M-Bus: Last and module count
    - RFID: Tags count

The Modem RSSI is displayed graphically indicating the numbers of bars:



The CSQ is displayed in raw format and in dBm.

Actions panel

The actions panel contains:

- A **Request** button: This button has the same effect as the physical request button on the gateway. A popup window will display all steps of the connection including the NTP synchronization, the checking of the Inbox and indicate all files that are uploaded.
- A **Reboot** button: This button will restart properly the gateway.
- An **M-Bus scan** button as described in paragraph 5.2.
- A **Wavenis RSSI scan** button that will retrieve the RSSI for all Wavenis modules. A popup window will display the RSSI values and a summary indicating the modules that were not reachable.
- A form to upload a configuration file or an updater to the gateway.

14.3 Local FTP Server

The gateway runs an FTP Server. It works as a local INBOX. Only one file can be uploaded at a time. It accepts configuration files and updater binaries.

15 Firmware upgrade

15.1 Local upgrade

An upgrade can be done using the web interface with a binary file previously downloaded.

15.2 Remote upgrade

An upgrade can be done remotely. The new firmware must be made available on the FTP server in the dedicated BIN directory. An update command must then be sent to the gateway.

Appendix A XML Schema / Configuration

See file config.xsd

Appendix B XML example / Configuration

See file config.xml

Appendix C XML Schema / Alarm

See file alarm.xsd

Appendix D XML example / Alarms

See file alarm.xml

Appendix E XML Schema / Supervision data

See file supervision.xsd

Appendix F XML example / Supervision data

See file supervision.xml

Appendix G XML Schema / Data

See file data.xsd

Appendix H CSV format / Data

The CSV format (Comma Separated Values) is a format with no formal definition.
Nevertheless, it follows the rules:

- One line contains a single recording
- Each recording is only one line long
- Each line ends with a line return.
- Each line contains the same number of fields.
- Each field is separated by a comma.

Each line is formatted as follow:

<Timestamp>,<Source>,<Address>,,,<Label>,<Input>,<Value>

with:

- Timestamp: DD/MM/YYYY HH:MM
- Source: Possible values are:
  - FL: WaveFlow
  - TH: WaveTherm
  - LO: WaveLog
  - SE: WaveSense
- Address: Wavenis address
- Label: Wavenis module label if configured, empty otherwise.
- Input: Input identifier (A, B, C, or D).
- Value: Value for the given input.

Data obtained from a Wavenis module datalog query are always consecutive lines in the CSV file and never distributed between two CSV files.

Appendix I XML Schema / Command

See file command.xsd

Appendix J XML example / Command

See file command.xml