IntelliWeb external MQTT API

Mihaela Grădinaru
Martijn Maarse

1. Introduction

This is the IntelliWeb API publicly available for external applications. With the help of this API, monitoring data can be gathered from different type of devices (i.e. inverters, batteries, ac measurement devices) and commands can be sent to these devices.

2. Server address and connection type

Server address: intelliweb.mastervolt.com
Port: 8883
Connection type: TLS/SSL

2. Authentication and authorization for MQTT

The authentication and authorizations of the MQTT clients will be done by the MQTT broker.

Authentication

On connect, each client will send a username and password and the broker will authenticate the client using that.

Each external MQTT client will be associated with an already existing user account from IntelliWeb. A user which wants to grant access to it's account via the API can create an API key in the IntelliWeb web interface.

The user account is identified by it's email address and this address will be used as both *MQTT client ID* and as *MQTT username*.

The password will be the API key of the client.

Authorization

Authorization is implemented in the broker.
Each external client is only allowed to write and subscribe to topics containing their own account id's in the topic name.

This way, a client with ID john@mail.com will only have access to topics matching this pattern:
/IntelliWeb/v1/account/john@mail.com/#

The IntelliWeb server will make sure that messages from all devices belonging to an user account will be cumulated in this topic structure.

Further, the clients will only be allowed to write to the "*sendCommand*" topic and to subscribe to the other topics.

Certain characters are not allowed in an MQTT topic that *are* allowed in e-mail addresses, namely '#', '+', and '/'. To that end, e-mail addresses are escaped as follows:

- + characters are replaced by &plus
- # characters are replaced by &hash
- pro/ characters are replaced by &slash
- & characters are replaced by &amp

Also, the e-mail addresses are always converted to lower case.
For example, an e-mail address 'John+special@mail.com' will result in a topic pattern:
/IntelliWeb/v1/account/john&plusspecial@mail.com/#

Demo account

An exception to the authentication described is the 'demo' acount.
The demo account can be accessed by connecting as user "demo" withour a password. The topic pattern is:
/IntelliWeb/v1/account/demo/#

3. MQTT Topics

External applications will be able to communicate with devices via the IntelliWeb server through MQTT topics, defined as follows:

1. topics used on which data from the connected devices will be posted:
    a. /IntelliWeb/v1/account/<account_id>/devices/<device_id>/status
    b. /IntelliWeb/v1/account/<account_id>/devices/<device_id>/monitoringData
    c. /IntelliWeb/v1/account/<account_id>/devices/<device_id>/commandCapability
    d. /IntelliWeb/v1/account/<account_id>/devices/<device_id>/commandResponse
2. topics that the are used to control the connected devices:
    a. /IntelliWeb/v1/account/<account_id>/devices/<device_id>/sendCommand

The account id will be defined by the email address of the user account.

1.1. Publishing topics

The IntelliWeb server will publish on the following topics.

1.1.1. The *status topic*

This topic will be used to update the external systems on the latest state of the device. A new message will be sent only when the status changes.
The payload of the *status* topic's messages will be in JSON format using the UTF-8 character set, looking like this:

{ "Timestamp": <Unix Timestamp (ms)>,

“Firmware”: <firmware version>,
“Status”: <Status> ,
“Errors”: [{“Timestamp”: <Timestamp>, ”ErrorCode”: <Error Code>},…
}
This payload structure is the same for each device type.

Explanation of values:

- Timestamp: the time the monitoring data was generated;
- Firmware: firmware version of the device;
- Errors: list of error codes
  - Timestamp:
  - ErrorCode: Standard error code from the Mastervolt Error code list [TO DO: Link]
- Status: status of the inverter with following possible values:
  - “Starting_up” → Preparing for normal operation;
  - “On” → Normal operation;
  - “On_warning” → Converting power, but a warning condition applies (i.e. power derating due to high temperature);
  - “Off_fault” → Indicating a condition where no power conversion is done due to environmental circumstances, i.e. grid problems, temperature out of bounds etc.;
  - ”Off_hardware_error” → Indicates a device which is broken.

1.1.2. The m*onitoringData* topic

This topic is used for publishing monitoring data to the outside world. The frequency of messages will be determined using the *setInterval* command.
The payload of the *monitoringData* topic’s messages will be in JSON format using the UTF-8 character set and it will differ per device type.

Solar Inverter

For an inverter, the payload will be defined like this:

{ “Timestamp”: <Unix Timestamp (ms)>,
“Dc”: [{“DCInID”: <DCInputId>, “V”: <Voltage (V)>, “I”:<Current(A),
“P”:<Power(W)} ,...],
“Ac”: [{“Phase”:<Phase number (1..3)>, “V”: <Voltage (V)>, “I”: <Current (A)>,
“P”: <Power (W)>}, …] ,
“Fac”: <Frequency (Hz)>,
“E”: <Total Yield (kWh)>,
“T”: <Temperature (C)>
}
Explanation of values:

- Timestamp: the time the monitoring data was generated;
- DC: list of DC input descriptors (one for each string of solar panels connect to the inverter)

    - DCInID: ID of the string of panels. (1..2)
    - V: input voltage (resolution 0,1 V) (0..1200)
    - I: input current (resolution 0,01 A, optional) (0..100)
    - P: input power (resolution 1W, optional)(0..50000)
- AC: list of AC outputs containing either single element for single phase inverters or three elements for three phase inverters
    - V: output voltage (resolution 0,1 V)(0..300) (typically around 230)
    - I: output current (resolution 0,01 A)(0..200)
    - P: output power (resolution 1 W)(0..50000)
- Fac: frequency of AC grid, resolution 0,01 Hz (0..70) (typically very close to 50);
- E: total yield generated since the inverter has been installed (resolution 0,01 kWh)(0..∞);
- T: temperature of the inverter (resolution 1 ˚C)(-50..100)(typically.. well depends on weather and inverter load)

AC power analyser (storage mode)

For an AC power analyser in storage mode, the payload will look like this. Field in *italics* are optional and might not be sent by all devices

```
{
        “Timestamp”: <UTC Timestamp (ms)>,
         “AcGrid”: [{
                “Phase”:<Phase number (1..3)>,
                “V”: <Voltage (V)>,
                “I”:  <Current (A)>,
                “P”: <Power (W)>,
                “F”: <Frequency (Hz)>}, ...
                “CosPhi”: <Cos φ>,
                “E_in”:  <Total Feed-in from grid (kWh)>,
                “E_out”:  <Total Feed-out to grid(kWh)>
                }] ,
        “Battery”: {
                “SOCRaw”:<Raw SOC (%)>,
                “SOCUser”:<User SOC (%)>,
                “Dc”: {
                        “V”: <Voltage (V)>,
                        “I”:<Current(A)>
                        “P”:<Power(W)
                        }
                “E_in”:  <Total charge energy (kWh)>,
                “E_out”:  <Total discharge energy (kWh)>
                },
        “SolarInverter”:{
```

"P": <Power (W)>,
"E_out": <Total Yield (kWh)>
}
"Load":{
"P": <Power (W)>,
"E_in": <Total Consumption(kWh)>
}
}

Explanation of values:

- Timestamp: the time the monitoring data was generated;
- ACGrid: measurements on the exchange between the system and the power grid
  - Phase: AC line phase 1..3 integer
  - V: voltage (0 .. 300) double
  - I: current (-1000 .. 1000) double
  - P: output power (resolution 1 W)(-230000 .. 230000)
  - CosPhi: cos phi (resolution 0,01)(0 .. 1)
  - E_in: Total Feed-in from grid (kWh) (>= 0)
  - E_out: Total Feed-out to grid(kWh) (>= 0)
- Battery: measurements regarding the storage of electrical energy
  - "SOCRaw": Gross SOC in which 0% means 0 Ah left (0 .. 100 %) (double),
  - "SOCUser": Net SOC, in which 0% means no more energy can be used without damaging the battery (0 .. 100 %)(double)
  - "Dc": {
    - "V": <Voltage (V)> 0 .. 100 double
    - "I":<Current(A)> -200 .. 200 double
    - "P":<Power(W)> -20000 .. 20000 double
  - "E_in": <Total charge energy (kWh)> (>= 0) double
  - "E_out": <Total discharge energy (kWh)> (>= 0) double
- SolarInverter
  - P: <Power (W)>, (0 .. 20000) double
  - E_out: <Total Yield (kWh)(>= 0) double
- Load
  - P: <Power (W)>, (-20000 .. 0) double
  - E_in: <Total Consumption(kWh)(>= 0) double

1.1.3. The *commandCapability* topic

This topic is used for communicating to the outside world what the capabilities of the devices are at the moment (related to the commands that the device can handle). A new message will be sent when the capabilities have changed.

The payload of the c*ommandCapability* topic's messages will be in JSON format using the UTF-8 character set, looking like this:

{ “Timestamp”: <Unix Timestamp (s)>,
“CommandName”:<Command>,
”Ranges”: [{“<parameter_name>”:[<Min>,<Max> OR N/A]} ,...]
}

The possible command names and ranges are defined in the table below:

	Param1 Range	Param2 Range	Solar Inverter	Storage	ACSwitch
disconnectGrid					X
connectGrid					X
batteryChargeRate	chargeRate <-100%, +100%>			X	
limitOutputPower	outputPower <0, 100>		X		
setInterval	interval <1, 3600>		X	X	
updateFirmware	updateAvailable (0, 1)		X	X	X

1.1.4. The *commandResponse* topic

This topic is used for posting the responses to the commands already sent to the devices. A new message will be sent for each message already sent containing a command.

The payload of the *commandResponse* topic’s messages will be in JSON format using the UTF-8 character set, looking like this:

{ “Timestamp”: <Unix Timestamp (s)>,
“OriginalCommandID”: <commandID>,
“Status”: <0, for success; -1, for failure>,
“Error”:<error_message>,
”ReturnedValues”: [{“<ParameterName>”: <param_value>},...]

}

Explanation of values:

- Timestamp: the time the message data was generated;
- OriginalCommandID: this message is a response to another mqtt message, which contained the command. This is the command ID of the that original message, so that the correlation can be made between the command and the response to that command;
- Status: 0, for success; -1, for failure; if a command failed, the server needs to know;
- Error: <error_message>; if an error occurred, the device needs to tell the server what went wrong.
- Some commands, have also returned values:
    - "updateFirmware" has 2 retuned values:
        - Firmware: The current firmware version;
        - Trigger: Was the updated triggered by the server ("server") or locally ("local")?

1.1.6. Subscribing to topics

An external system will be able to subscribe to a specific topic of a specific account or will use wildcards to subscribe to multiple topics, like this:

/IntelliWeb/v1/account/john@mail.com/devices/130001500_S100A0001/status - status topic of the user account with email john@mail.com for device article number 130001500 and serial number S100A0001.

/IntelliWeb/v1/account/john@mail.com/# - all topics of the user account with email john@mail.com

/IntelliWeb/v1/account/john@mail.com/+/monitoringData - monitoring data of all devices of the user account with email john@mail.com

The authorization mechanism will actually limit the access to topics. Since authorization will be user account based, an external system will only be allowed to subscribe to topics belonging to the devices associated with the given account. That is why a client can not use a wildcard for the account part of the topic name. So the minimum part that needs to be explicit is */IntelliWeb/v1/account/<account_id>/*

1.2. Subscribing topics

The IntelliWeb server will also subscribe to a set of topics. This way, the external systems can communicate with the server also.
These topics are specified as follows.

1.2.1. The *sendCommand* topic

This topic is used by the external systems to send commands to the devices. For now we have a limited set of supported commands.
The expected payload of the *sendCommand* topic's messages will be in JSON format using the UTF-8 character set, looking like this:

```
{       "Timestamp": <Unix Timestamp (s)>,
        "CommandID": <commandID>,
        "CommandName":<Command>,
         "Parameters": [ {"<parameter_name":<param_value>} ,... ]
}
```

The "CommandID" field uniquely identifies the command on the server. This is used for correlation of the command itself and the response for this command.

	Param1	Param2
disconnectGrid		
connectGrid		
batteryChargeRate	chargeRate=80	
limitOutputPower	outputPower=80	
setInterval	interval=5	
updateFirmware		

Command setInterval

This topic will be used by the external systems to tell to the device how often it should send monitoring data.

Each inverter must post monitoring data every <interval>.
If the <interval> is changed, the inverter must take this into account immediately. For example:
11:00:00 inverter sends data, interval = 600s
11:01:00 inverter receives new interval of 10s via *setInterval* message
11:01:00 as the time passed since the last message (11:00:00) is larger than the current interval (10s),

the inverter sends out a new monitoring message immediately (and repeats every 10s)
11:03:00 inverter receives new interval of 600s *setInterval* message
11:03:10 nothing happens: current interval is 600s
11:13:00 the time passed since the last message exceeds the interval again, the inverter sends out a new monitoring message

When it's not possible to post a status update, the update will be stored on the inverter in a rotating buffer, large enough to contain at least 1500 status records. This buffer will be retained when the inverter powers off.
The inverter will try to get a connection again once the interval has passed. Once the inverter has successfully posted a message to the server, the inverter removes the message from the queue and will continue to send the next message in queue, until the queue is empty.

TBD: do we still want a queue size?

Command updateFirmware

This topic is used by the server to communicate to the external systems that a firmware update is available and to trigger the update. The IntelliWeb knows the details of the update, like url where is can be found.

4. Quality Of Service

All the MQTT messages will be sent using a QOS value of 2, meaning they will be delivered exactly once.