



MQTT PROTOCOL USER MANUAL

Index

Introduction	3
1. Plant configuration.....	3
2. Configuration and initial mapping.....	3
3. Topics	3
1 MQTT Service Description	5
2 Telemetry messages	6
2.1 Complete message.....	6
2.2 Normal message.....	6
2.3 Jsn Time series (JTS).....	7
3 Queries to IoT Scada	8
3.1 System Information.....	8
3.1.1 Information about IoT Scada.....	8
3.2 Configuration of connected devices	9
4 Variables Information	10
4.1 Configuration of the variables.....	10
4.2 Current data of the variables	11
4.3 Log data.....	12
4.4 Setting a variable.....	13
5 Variables Information	14
5.1 Configuration of the alarms	14
5.2 Alarms status.....	15
6 Events Information	16
6.1 Configuration of the events.....	16
6.2 Historical data of the events	17
7 Other automatic messages	18
7.1 New alarm raised.....	18
7.2 New event raised.....	19

Introduction

1. Plant configuration

The gateway publishes the data of the devices, monitored on the customer's MQTT broker.

Once the Alleantia software has been set up in the field, the broker addressing and the frequency of sending data are entered, the system will automatically send them.

It will also be necessary to select the variables, alarms and events beforehand to be forwarded through the MQTT service.

2. Configuration and initial mapping

First, the configuration and mapping of the system must be scanned, in order to recreate its state:

- the number of gateways in the field: each of them will have its own Serial Id.

For each gateway in the field, first query phase is necessary to create a lookup table to keep the data decryption keys.

These queries are reported in Section 3.2 onwards in this manual, in order to understand and to map:

- the number of devices in the field and their unique nomenclature (dev Id): this is the number of equipment/machines monitored by the specific software license;
- the number of variables that underlie each device (var id): these variables are the data such as speed, temperature, etc., transmitted by the machine.

Once the mapping of the system has been rebuilt, only the receipt of the value of the variable (value) is needed, which is communicated automatically by the device in the **Telemetry Topic**.

3. Topics

The data published by the system are divided into *Topics*.

Each gateway will publish messages on 3 different Topics:

- Topic Alarms
- Topic Events
- Topic Telemetry

The initial configuration has a fourth Topic, which the gateway listens to and waits for requests from the outside, according to the syntax (see Section 3):

- Topic Commands.

Each Topic has the following pattern: *Serial Id/Telemetry*, *Serial Id/Alarm*, *Serial Id/Events*, *Serial Id/Commands*.

Therefore, the Topic Commands will be used in order to realize the initial requests from the user, to rebuild the lookup of the system.

All the responses from the gateway will take place on the publication Topics mentioned above, especially the Topic Telemetry.

Introduction

Only information related to alarms or events, set in the system (Alleantia software) will be published on the related **Topic Alarms** and **Events**.

Note:

Every addition in the field of monitored devices and variation of the configuration implies updating the lookup and the replication of requests for information to the **Topic Commands**.

Final summary

First, make connection to the gateways in the field in order to determine which machines are monitored and which variables each machine communicates.

Following the mapping of the system it will be possible to read the values of the variables whose origin (machine and gateway) will be known.

There is a set of initial calls on the **Topic Commands** and the gateway will respond on the **Topic Telemetry**.

The automated information on 3 Publication **Topics** will be sent regularly.

1 MQTT Service Description

IoT Scada provides the message forwarding service to the cloud, using MQTT protocol.

If the license is enabled and if the service has been activated, IoT Scada will regularly send the telemetry messages as a string in JSON format to configured MQTT broker.

The IoT Scada provides a mechanism for configuring permissions (read/write/forward alarms) of the variables to forward, therefore only the variables with such permission will be forwarded through the telemetry messages. Even the writing operation (if logically possible) will be carried out only if such permission has been previously enabled in the configuration.

There are three types of messages that can be forwarded automatically from the device, which differ in the amount of provided information and the syntax (and therefore in message size).

In the IoT Scada, every single variable, collected from the field, is uniquely identified by the **devId** (device identifier the variable belongs to) and **varId** (variable identifier). Some variables may have an unadjusted **devId**: these are so-called **Plant variables** (called **Custom Measures** in the IoT Scada), that do not have such identifier due to their nature: they do not belong to any device, but they are the result of an elaboration made on several variables, basically coming from different devices (example: the sum of kWh read by N inverters connected to the IoT Scada).

The operating principle ensures that only the minimum amount of data is forwarded, provides that a first, only one query is performed during initialization, by forwarding one or more messages in JSON format (explained later in this manual) to the broker, to get as a response the device configuration, connected devices, configuration of variables, alarms, events, and to build a lookup table with them.

In this way, the user will have the keys to identify the data forwarded automatically in the subsequent telemetry messages, so that this information should not be forwarded again in every message.

2 Telemetry messages

The telemetry messages, automatically forwarded with the frequency, that can be set in the device interface, have the following syntax:

2.1 Complete message

```
{
  "telemetryDataList":[
    {
      "date":"Jan 8, 2018 10:50:30 AM", //java.util.Date
      "category":"main", //String
      "blockId":"B2", //String
      "description":"Ingresso digitale 3, overflow conteggio parziale",
      "deviceDescription":"IoT server-11x",
      "dataType":"BOOLEAN", //String
      "devId":3, //int
      "varId":171, //int
      "value":false, //Object(String,Boolean,Numeric)
      "quality":true //boolean
    }, //..... N variabili
  ],
  "devSn": "IOTSPIXXXXXXXXXX", //String
  "onTime": " Jan 8, 2018 10:50:34 AM" //java.util.Date
}
```

2.2 Normal message

```
{
  "telemetryDataList":[
    {
      "date":"Jan 8, 2018 11:13:27 AM",
      "devId":3,
      "varId":171,
      "value":false,
      "quality":true
    }, //..... N variabili
  ],
  "devSn": "IOTSPIXXXXXXXXXX",
  "onTime": " Jan 8, 2018 11:13:27 AM "
}
```

2 Telemetry messages

The list of **telemetryData**, restored from IoT Scada, is a JSON consisting of:

- devId – identifier of the device, the variable belongs to (can be null in the case of plant variables)
- varId – identifier of the variable
- value – contains the value, read for the variable
- quality – boolean value, indicating whether last read attempt was successful or not, so if the value is up-to-date or potentially legacy
- date – date, last correct reading of the variable dates back to

Each message will have two header fields:

- devSn – contains the unique serial number of the device that forwards the variables
- onTime – forwarding date of the message

N.B. It is possible to distinguish these two types of messages (complete message and normal message) simply by ticking or not ticking in the IoT Scada interface “**The forwarded message will contain only the essential information**” entry.

2.3 Json Time series (JTS)

```
{
  "samples":5,
  "samplesDelay":0,
  "startTime":1515406966223,
  "endTime":1515406970224,
  "telemetryDataList":[
    {
      "devId":3,
      "varId":171,
      "values":[false, false, true, true, false],
      "quality":[0,1,1,1,1]
    }, //..... N variabili
  ],
  "devSn": "IOTSPIXXXXXXXXXX",
  "onTime": " Jan 8, 2018 11:13:27 AM "
}
```

The message contains the following header parameters:

- samples (s) – number of samples taken in the unit of time (can be set in MQTT Service interface);
- samplesDelay – Time interval of the single sampling (deriving from the formula $(sT-eT)/s$);
- startTime (sT) – start time of the sampling series;
- endTime (eT) – final timestamp of the sampling series.

N.B. It is possible to choose the following type of message, simply ticking “*JTS (Json Time Series)*” entry, in IoT Scada interface. After that, it is possible to set the number of samplings, carried out in the time interval (Samples).

3 Queries to IoT Scada

As mentioned above, the IoT Scada can listen while waiting for messages, published on the broker, and respond promptly to requests for information.

The IoT Scada device will listen to the broker *devSn/commands* and respond to the requests *devSn/telemetry*.

All messages, recognized by the IoT Scada, have the same basic structure, to be completed in a different way, depending on the type of request.

The message recognized by IoT Scada has the following structure:

```
{
  "component": "",
  "operation": "",
  "devId": [],
  "varId": [],
  "startTime": "",
  "endTime": "",
  "value": ""
}
```

Evidently, as it will be explained later, some fields must be filled in for certain requests, and left empty for others, in case of non-recognition such message and response.

The basic structure consists of:

- component – (compulsory) - list of the involved components (INFO, DEVICES, ALARMS, EVENTS);
- operation – (compulsory, except INFO component) - list of the possible operations (LIST, DATA, CONFIG, LOGDATA, SET, ACTIVE, HISTORY);
- devId – (optional) – list of deviceIds;
- varId – (optional) – list of the variables;
- startTime - (optional) – initial endpoint of the time interval to extract data;
- endTime - (optional) – final endpoint of the time interval to extract data;
- value - (optional) – value to write in the variable.

3.1 System Information

3.1.1 Information about IoT Scada

To get such information, the JSON message to forward is:

```
{
  "component": "INFO"
}
```


3 Queries to IoT Scada

The IoT Scada will respond with information related to it:

```
{
  "uuid": "49ab91753-9b94-3d2e-8ff6-17d3d5113606",
  "hwModel": "4.0.5-SNAPSHOT",
  "name": "IoT Gateway",
  "webAppVersion": "4.0.3",
  "devSn": " IOTSPIXXXXXXXXXX ",
  "onTime": "Sep 20, 2017 5:23:12 PM"
}
```

3.2 Configuration of connected devices

To get such information, forward the following JSON message:

```
{
  "component": "DEVICES",
  "operation": "LIST",
  "devId": [ 2, 3, 4 ] //OPTIONAL
}
```

The parameter list **devId** is optional, it filters the information: if this parameter is not included, the device will provide the information from all devices. If it is set, only the information from the selected ones will be returned.

The IoT Scada will respond with a series of information regarding the device:

```
{
  "devices": [
    {
      "devId": 2,
      "description": "Fanuc 750",
      "linked": true
    },
    {
      "devId": 3,
      "description": "Fanuc 545",
      "linked": true
    },
    {
      "devId": 4,
      "description": "MCM-cnc",
      "linked": false
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX ",
  "onTime": "Sep 20, 2017 5:23:12 PM"
}
```

4 Variables Information

4.1 Configuration of the variables

To get such information, forward the following JSON message:

```
{
  "component": "DEVICES",
  "operation": "CONFIG",
  "devId": [ 63 ], //OPTIONAL
  "varId": [ 40, 41, 39 ] //OPTIONAL
}
```

The parameter list **devId** is optional, it filters the information: if this parameter is not included, the device will provide the information from all devices. If it is set, only the information from the selected ones will be returned.

The parameter list **varId** is optional. If specified, there should be **only one** devId indicated in the corresponding parameter. The device will provide only the information regarding the selected varId for this device.

The IoT Scada will respond with a series of information regarding the device:

```
{
  "varConfigList": [
    {
      "devId": 63,
      "varId": 21,
      "description": "Actual executed program name",
      "dataType": "String",
      "minimum": "null",
      "maximum": "null",
      "category": [ "main" ],
      "alarmable": false,
      "writable": false
    },
    {
      "devId": 63,
      "varId": 23,
      "description": "Automatic mode selection",
      "dataType": "Numeric",
      "minimum": "null",
      "maximum": "null",
      "category": [ "main" ],
      "alarmable": false,
      "writable": false
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 4:11:13 PM"
}
```

4 Variables Information

4.2 Current data of the variables

To get such information, forward the following JSON message:

```
{
  "component": "DEVICES",
  "operation": "DATA",
  "devId": [ 63 ], //OPTIONAL
  "varId": [ 23, 430 ] //OPTIONAL
}
```

The parameter list **devId** is optional, it filters the information: if this parameter is not included, the device will provide the information from all devices. If it is set, only the information from the selected ones will be returned.

The parameter list **varId** is optional. If specified, there should be **only one** devId indicated in the corresponding parameter. The device will provide only the information regarding the selected varId for this device.

The IoT Scada will respond with a series of information regarding the device:

```
{
  "variablesList": [
    {
      "devId": 63,
      "varId": 23,
      "value": "MDI",
      "decodedValue": "MDI",
      "date": "Sep 20, 2017 5:48:57 PM",
      "quality": false
    },
    {
      "devId": 63,
      "varId": 430,
      "value": false,
      "date": "Sep 20, 2017 5:48:57 PM",
      "quality": false
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 4:29:46 PM"
}
```

4 Variables Information

4.3 Log data

To get such information, forward the following JSON message:

```
{
  "component": "DEVICES",
  "operation": "LOGDATA",
  "devId": [ 63 ], //COMPULSORY
  "varId": [ 23 ], //COMPULSORY
  "startTime": 0, //OPTIONAL
  "endTime": 123456789 //OPTIONAL
}
```

The parameters list **devId** are compulsory. In every list there must be only one value. The parameters **startTime** and **endTime** are optional. These parameters apply a temporary filter on the data to be restored. The values to be entered are expressed in milliseconds starting from 1/1/1070 0:00 GMT.

The IoT Scada will respond with a series of information regarding the variables:

```
{
  {
    "devId": 63,
    "varId": 23,
    "value": "MDI",
    "date": "Sep 8, 2017 1:37:14 PM",
    "quality": true
  },
  {
    "devId": 63,
    "varId": 23,
    "value": "MDI",
    "date": "Sep 8, 2017 1:40:14 PM",
    "quality": true
  }
],
"devSn": " IOTSPIXXXXXXXXXX",
"onTime": "Feb 20, 2017 5:02:49 PM"
}
```

devId and **varId** fields are optional. If left blank, the IoT Scada will respond with the list of all the variables; if one or more **devIds** are specified, the output will be filtered, returning only the requested ones.

If also a list of **varIds** is specified, the **devId** must be unique: only the variables with such **devId** and **varId** will be returned.

4 Variables Information

4.4 Setting a variable

To get such information, forward the following JSON message:

```
{
  "component": "DEVICES",
  "operation": "SET",
  "devId": [ 31 ], //COMPULSORY
  "varId": [ 47 ], //COMPULSORY
  "value": 10 //COMPULSORY
}
```

The parameters list **devId** are compulsory. In every list there must be only one value. **Value** parameter indicates the value of the variable to be set.

The IoT Scada will respond with a series of information regarding the variables:

```
{
  "accepted": true,
  "description": "Accepted",
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:02:49 PM"
}
```

5 Variables Information

5.1 Configuration of the alarms

To get such information, forward the following JSON message:

```
{
  "component": "ALARMS",
  "operation": "CONFIG",
  "varId": [ 47, 48] //OPTIONAL
}
```

The parameter list **varId** is optional, it filters the information: if this parameter is not included, the device will provide the information from all alarms. If it is set, only the information from the selected ones will be returned.

The IoT Scada will respond with a series of information regarding the alarms:

```
{
  "alarmConfigList": [
    {
      "id": 48,
      "description": "BULBING TAIL ALARM",
      "condition": "$G_63_86 eq true"
    },
    {
      "id": 43,
      "description": "DRIVEN TOOLS ALARM",
      "condition": "$G_63_81 eq true"
    },
    {
      "id": 47,
      "description": "CLIP ALARM",
      "condition": "$G_63_85 eq true"
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```

5 Alarms Information

5.2 Alarms status

To get such information, forward the following JSON message:

```
{
  "component": "ALARMS",
  "operation": "DATA",
  "varId": [ 47, 48] //OPTIONAL
}
```

The parameter list **varId** is optional, it filters the information: if this parameter is not included, the device will provide the information from all alarms. If it is set, only the information from the selected ones will be returned.

The IoT Scada will respond with a series of information regarding the alarms:

```
{
  "alarmDataList": [
    {
      "id": 48,
      "quality": true,
      "alarmed": false
    },
    {
      "id": 43,
      "quality": true,
      "alarmed": true
    },
    {
      "id": 47,
      "quality": false,
      "alarmed": false
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```

6 Events Information

6.1 Configuration of the events

To get such information, forward the following JSON message:

```
{
  "component": "EVENTS",
  "operation": "INFO",
  "varId": [ 1 ] //OPTIONAL
}
```

The parameter list **varId** is optional, it filters the information: if this parameter is not included, the device will provide the information from all events. If it is set, only the information from the selected ones will be returned.

The IoT Scada will respond with a series of information regarding the events:

```
{
  "eventsInfoList": [
    {
      "eventId": 1,
      "eventName": "New Event",
      "type": "boolean",
      "condition": "$G_63_71",
      "snapshotGlobalIds": "G_63_21",
      "comparisonOperator": "eq",
      "numericCompareValue": 1
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```


6 Events Information

6.2 Historical data of the events

To get such information, forward the following JSON message:

```
{
  "component": "EVENTS",
  "operation": "HISTORY",
  "varId": [ 1 ],      //COMPULSORY
  "startTime": 0,     //OPTIONAL
  "endTime": 123456789 //OPTIONAL
}
```

The parameter list **varId** is compulsory. There must be only one value in the list. **startTime** and **endTime** parameters are optional. These parameters apply a temporary filter on the data to be restored. The values to be entered are expressed in milliseconds starting from 1/1/1070 0:00 GMT.

The IoT Scada will respond with a series of information regarding the events:

```
{
  "eventHistoryList": [
    {
      "eventId": 1,
      "eventName": "New Event",
      "timestamp": "Sep 11, 2017 11:00:58 AM",
      "state": true,
      "variablesSnapshot": [
        {
          "devId": 63,
          "varId": 21,
          "value": "//CNC_MEM/USER/PATH1/TECNO",
          "quality": true
        }
      ]
    },
    {
      "eventId": 1,
      "eventName": "New Event",
      "timestamp": "Sep 11, 2017 11:11:28 AM",
      "state": true,
      "variablesSnapshot": [
        {
          "devId": 63,
          "varId": 21,
          "value": "//CNC_MEM/USER/PATH1/O9110",
          "quality": true
        }
      ]
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```

7 Other automatic messages

Once MQTT service has been activated on the IoT Scada, it will be able to forward automatically to the broker, besides the telemetry messages, messages regarding new alarms and/or events, when they occur.

Evidently, as for the variables, in order for the service to report them, even any custom alarms and/or events must be enabled to forward, in the IoT Scada interface.

The syntax of the automatically forwarded messages are as follows.

7.1 New alarm raised

In case of a new alarm, the following JSON message will be forwarded to the topic, configured for receiving alarms:

```
{
  "activeAlarmsList": [
    {
      "id": 11,
      "eventId": 16153,
      "deviceName": "Fanuc 545",
      "measure": "GUARD OPEN",
      "description": "GUARD OPEN",
      "onDate": "Sep 22, 2017 10:08:09 AM",
      "offDate": "Sep 22, 2017 10:10:25 AM"
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```

The message will be forwarded twice: when the alarm is activated, and therefore the parameter **offDate** will be absent, and when it returns, and it will be compiled as shown above.

7 Other automatic messages

7.2 New event raised

In case of a new event, the following JSON message will be forwarded to the topic, configured for receiving events:

```
{
  "newEventsList": [
    {
      "eventId": 1,
      "eventName": "New Event",
      "type": "boolean",
      "condition": "$G_63_71",
      "snapshotGlobalIds": "G_63_21,G_63_820",
      "comparisonOperator": "eq",
      "numericCompareValue": 1,
      "timestamp": "Sep 21, 2017 2:58:49 PM",
      "snapshotVarsDatas": [
        {
          "globalId": "G_63_21",
          "snapshotValue": "//CNC_MEM/USER/PATH1/TECNO"
        },
        {
          "globalId": "G_63_820",
          "snapshotValue": false
        }
      ],
      "eventValue": "true"
    }
  ],
  "devSn": " IOTSPIXXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```

Two types of events exist: **boolean**, triggered when a condition occurs, and **onChange**, triggered when the value of a variable has changed.

For **boolean** type of event, two messages will be forwarded to the broker: one when the condition occurs, and another when the condition is no longer true.

OnChange events, by their nature, will forward a message to the broker whenever the variable they monitor changes its value.



Alleantia s.r.l.

www.alleantia.com

R.O.: Via Tosco Romagnola, 136 - 56025 Pontedera (PI), Italy
H.Q.: Via Umberto Forti, 24/14 56121 Pisa, Italy

VAT code: IT 02011550502

Tel: (+39) 050 9911933
Fax: (+39) 050 9655139
@: sales@alleantia.com

M-MQTT-0618-EN