Table of Contents

www.alleantia.com

1 Azure IoT Hub Service Description

IoT SCADA provides a service of message sending to the cloud through Microsoft Azure IoT Hub.
If the license has been enabled and the service has been activated, IoT Scada will periodically send telemetric messages as JSON format strings to the configurated IoT Hub.
IoT Scada provides a mechanism of configuration of permits (writing/reading/sending alarms) relative to the variables that are needed to be sent. Therefore, only variables for which permit is already been given will be sent through telemetric messages. The writing operation (if logically possible) will also be carried out only if the respective permit was enabled in the configuration.
At present there are two types of automatically sendable messages from the device. The two types differ only in the number of information that they contain, for which only in the final size of the message that needs to be sent.

In IoT Scada every single variable that has been collected from the field is uniquely identified by the couple devId (identification of the variable's device) and varId (identification of the variable), It's possible that some variables have a not-valorised devId: these are the so-called field variables (called also Custom Measures in IoT Scada). These variables do not have this type of identification give the way they are created: they do not belong to any device but are the result of an elaboration made on many variables, usually from different devices (ex.: the sum of kWh read from N inverters connected to IoT Scada),

Microsoft Azure IoT Hub expects the payment of licenses with variable prices based on the number of daily messages sent to the Hub. Currently the transported payload of any single message is 0,5KB for the free trial version and 4KB for the payed version. It's

www.alleantia.com

therefore important to transfer the maximum quantity of data in the least space possible. For doing so, a mechanism has been created that needs the following operations:

The final user of the data executes a first and sole interrogation sending messages with a well-defined syntax (explained in the following chapters of this manual) to the Hub and so to IoT Scada to obtain as a reply the device's configuration, connected devise, variables' configuration, alarms and evens.
Subsequently, after having elaborated this information, the user will have access to the keys to uniquely identify the automatically sent data in the following telemetric messages. In this way the quantity of data to send for every single variable is reduced to a minimum.

www.alleantia.com

2 Telemetric messages

Telemetric messages automatically sent to a settable frequency in the device's interface has the following syntax:

2.1 Full Message

```
{
  "telemetryDataList": [
    {
      "date": "Jan 27, 2017 10:04:11 AM", //java.util.Date
      "devId": 31,  //int
      "varId": 39,  //int
      "value": 0,    //Object(String,Boolean,Numeric)
      "quality": false,  //boolean
    }, //..... N variables
  ],
  "devSn": "IOTSPIXXXXXXXXX",  //String
  "onTime": "Jan 27, 2017 10:04:11 AM"  java.util.Date
}
```

2.2 Data Saving

```
{
  "telemetryDataList": [
    {
      "devId": 31,
      "varId": 39,
      "value": 0,
      "quality": false,
    }, //..... N variables
  ],
  "devSn": "IOTSPIXXXXXXXXX",
  "onTime": "Jan 27, 2017 10:04:11 AM"
}
```

The list of telemetryData returned by IoT Scada are JSON objects composed by:

- devId – identification of the variable's device (can be null in the case of field variables)

www.alleantia.com

- varId – variable identification
- value – read value of the variable
- quality – boolean value that indicates if the last reading trial was successful, so if the content's value is updated or potentially outdated
- date – variable last successful reading date

Every message will then have two fields:

- devSn –contains the unique serial of the device that is sending variables.
- onTime – date of the message sending.

N.B. It's possible to discriminate between these two types of messages simply by checking or not *"The sent message will contain only essential information"* in IoT Scada interface.

www.alleantia.com

3 Interrogations to IoT Scada

As previously wrote, IoT Scada can receive sent messages from the hub, respecting a given syntax, and promptly responds to said information requests. Every message recognised by IoT Scada share the same basic structure, to be compiled differently in relation to the type of request.

The basic message recognised by IoT Scada is structured as following:

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

Obviously, as it will be explained later, some field will have to be compiled for specific calls and left blank for others. Otherwise the message will not be recognized, and no reply will be returned.

The basic structure is composed as follow:

- component – (mandatory) enumeration of the involved components (INFO, DEVICES, ALARMS, EVENTS)
- operation – (mandatory except for INFO component) enumeration of the possible operations (LIST, DATA, CONFIG, LOGDATA, SET, ACTIVE, HISTORY)
- devId – (optional) list of deviceIds
- varId – (optional) list of variables
- startTime – (optional) starting point of the temporal range from which to extract data
- endTime - (optional) ending point of the temporal range from which to extract data
- value - (optional) value to write on the variable

www.alleantia.com

3.1 System Information

3.1 IoT Scada Information

To obtain such information, the JSON message to send is:

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

IoT Scada will return information about IoT Scada:

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

www.alleantia.com

3.2 Connected Devices Configuration

To obtain such information, the JSON message to send is:

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

.

The devId list parameter is facultative and can be used to filtrate information. If such parameter is omitted, the device will return information of every device. By setting it, only information about those selected will be returned.

IoT Scada will return information about 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": " IOTSPIXXXXXXXXX ",
  "onTime": "Sep 20, 2017 5:23:12 PM"
}
```

www.alleantia.com

4 Variables Information

4.1 Variables Configuration

To obtain such information, the JSON message to send is:

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

The devId list parameter is facultative and can be used to filtrate information. If such parameter is omitted, the device will return information of every device. By setting it, only information about those selected will be returned.
The varId list parameter is facultative. If it is specified, there must be **only one** devId set in the correspondent parameter. The device will provide only the information relative to the selected varId for such device.

IoT Scada will return information about the variables:

```
{
  "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",
```

www.alleantia.com

```
      "dataType": "Numeric",
      "minimum": "null",
      "maximum": "null",
      "category": [ "main"    ],
      "alarmable": false,
      "writable": false
  ],
  "devSn": " IOTSPIXXXXXXXXX",
  "onTime": "Feb 20, 2017 4:11:13 PM"
}
```


www.alleantia.com

4.2 Real-time Variable Data

To obtain such information, the JSON message to send is:

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

The devId list parameter is facultative and can be used to filtrate information. If such parameter is omitted, the device will return information of every device. By setting it, only information about those selected will be returned.

The varId list parameter is facultative. If it is specified, there must be **only one** devId set in the correspondent parameter. The device will provide only the information relative to the selected varId for such device.

IoT Scada will return information about the variables:

```
{
  "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": " IOTSPIXXXXXXXXX",
  "onTime": "Feb 20, 2017 4:29:46 PM"
}
```

www.alleantia.com

4.3 Log Data

To obtain such information, the JSON message to send is:

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

The devId and varId list parameters are mandatory. There must be only one value in each list. The startTime and endTime parameters are facultative. Such parameters apply a time filter on the data to return. The value that need to be inserted are expressed in milliseconds starting from 1/1/1070 0:00 GMT.

IoT Scada will return information about 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": " IOTSPIXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:02:49 PM"
}
```

The devId and varId field are facultative. If they are left blank, IoT Scada will return the list of every variable. If one or more devIds are specified, the final output will be filtered

www.alleantia.com

returning only the ones requested. If a list of varId is also specified, the devId will need to be unique: only the variables with such devId and varId will be returned.

4.4 Variable Setting

To obtain such information, the JSON message to send is:

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

The devId and varId list parameters are mandatory. There must be only one value in each list. The value parameter indicates the value for which you want to set the variable.

IoT Scada will return information about the variables:

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

www.alleantia.com

5 Alarm Information

5.1 Alarm Configuration

To obtain such information, the JSON message to send is:

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

The varId list parameter is facultative and can be used to filtrate information. If such parameter is omitted, the device will return information of every alarm. By setting it, only information about those selected will be returned.

IoT Scada will return information about the alarms:

```
{
  "alarmConfigList": [
    {
      "id": 48,
      "description": "TAILSTOCK ALARM",
      "condition": "$G_63_86 eq true"
    },
    {
      "id": 43,
      "description": "MOTOR ALARM",
      "condition": "$G_63_81 eq true"
    },
    {
      "id": 47,
      "description": "CHARGER ALARM",
      "condition": "$G_63_85 eq true"
    }
],
  "devSn": " IOTSPIXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```

www.alleantia.com

5.2 Alarms State

To obtain such information, the JSON message to send is:

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

The varId list parameter is facultative and can be used to filtrate information. If such parameter is omitted, the device will return information of every alarm. By setting it, only information about those selected will be returned.

IoT Scada will return information about the alarms:

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

www.alleantia.com

6 Event Information

6.1 Event Configuration

To obtain such information, the JSON message to send is:

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

The varId list parameter is facultative and can be used to filtrate information. If such parameter is omitted, the device will return information of every event. By setting it, only information about those selected will be returned.

IoT Scada will return information about the events:

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

www.alleantia.com

6.2 Event Historical Data

To obtain such information, the JSON message to send is:

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

The devId list parameter is mandatory. There must be only one value in the list. The startTime and endTime parameters are facultative. Such parameters apply a time filter on the data to return. The value that need to be inserted are expressed in milliseconds starting from 1/1/1070 0:00 GMT. IoT Scada will return information about 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
        }
      ]
```


www.alleantia.com

```
    },
  ],
  "devSn": " IOTSPIXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```


www.alleantia.com

7 Other Automatic Messages

When the Azure IoT Hub service is activated on IoT Scada, it will be able to automatically send to the hub, telemetric messages and messages relative to new alarms and/or events in the moment when they happen.
Obviously, as with variables, to make possible that the service can notify them, eventual custom alarms and/or events need to be enabled to be sent to IoT Scada interface.
The syntax to recognize such messages are the following:

7.1 New Alarm Relieved

In the eventuality of a new alarm, the following JSON message will be sent to the hub:


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


The message will be sent twice: in the moment of the alarm activation, without the offDate parameter, and in the moment of its return, and will be compiled as seen above.

www.alleantia.com

7.2 New Event Relieved

In the eventuality of a new event, the following JSON message will be sent to the hub:


```
{
  "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": " IOTSPIXXXXXXXXX",
  "onTime": "Feb 20, 2017 5:25:48 PM"
}
```


There are two types of events: *boolean*, that are activated when one condition verifies, and *onChange,* activated by the changing of a variable value.

For the boolean event, two messages will be sent to the hub: one at the moment when the condition occurs, the other, at the moment when the condition is no more true.

The onChange events, given their nature, will send a message to the hub every time the value of the monitored variable changes.

www.alleantia.com