TSec MACS Rest API

Version: 1.1.0

TSec Srl

Abstract

This document describes the Rest API provided by MACS system

Contents

1 Overview

This document describes the Rest API interface of MACS, the anti intrusion perimeter system of TSec SpA. Communication is implemented by an HTTPS Rest API that allows reading/writing the device configuration and retrieve the device status.

The device communicates over a secure HTTP connection (HTTPS) and the client must install the proper certificate which is available at (TODO)

Most of the operation on the device are restricted to authenticated clients. A client needs to login to get authenticated. In order to login a client shall provide a username and a password and will get a session token that shall be added to any http call as an header in the form: **Authorization: Bearer token** where token is the verbatim token as received on login success.

Since tokens have an expiration date which is unknown the client must be ready to handle the unauthorized error (HTTP 403). If any call fails with such an error the client shall login and repeat the original call.

2 Command types

All commands are divided in read commands (HTTP POST request) and write commands (HTTP PUT request). Some commands share the same endpoint request but a special field in body (“VarName”) identify the specific scope of the packet.

2.1 Read commands

This commands are generally requested with the HTTP POST method.

2.1.1 Simple read request

Request some data with no specifications.

```
{
  "VarName": <String:v>
}
```

2.1.2 Read with structure request

Request some data of some element specified in “Data” field.

```
{
  "VarName": <String:v>,
  "Data": <JSONObj:js>
}
```

2.1.3 Read response

The response to a successful read command contains all the requested info. The formatting of the response depends on the type of infos.

2.2 Write command

These commands are generally requested with the HTTP PUT method.

2.2.1 Simple write request

Execute a command to set some simple data.

```
{
  "VarName": <String:value>,
  "Data": {
    "Value": <I or String:value>
  }
}
```

2.2.2 Write single value in multiple structures request

Execute a command applying same value to all the elements specified in the array. is the key used to describe the next JSONArray (e.g. “Sensors” if the JSONArray contains sensor descriptions).

```
{
  "VarName": <String:value>,
  "Data": {
    "Value": <I or String:value>,
    <SpecificKey>: <JSONArray:value>
  }
}
```

2.2.3 Write specific value in multiple structures request

Execute a command set different value for every element in JSON Array. The elements of JSON array can have multiple field but only the field linked to command is applied. is the key used to describe the next JSONArray (e.g. “Sensors” if the JSONArray contains sensor descriptions).

```
{
  "VarName": <String:value>,
  "Data": {
    "ValueObj": {
        <SpefificKey>: <JSONArray:value>
    }
  }
}
```

2.2.4 Write response

The response to a correctly formatted write command will be like that

```
{
  "Result": <I:value>
}
```

"Result" value contains the result of the command. Common values are:

- 0 = Operation executed succesfully
- 1 = Operation failed
- 2 = Operation failed due to wrong/forbidden data
- 3 = Operation failed due to wrong size of data
- 100 = Long sensor operation in progress, check system results through Last-Operation-Status packet
- 101 = Log preparation in progress, check system results through Log-Operation-Status packet

Other values may be used in response of some specific packets with a special meaning.

3 Data structures

3.1 Data structures description

This section will describe some common data structures used in MACS system. These structures can be used in read or write requests.

3.1.1 Sensor data structure

MACS system can handle at most 240 sensors. The sensors have a unique id from 1 to 120 if the sensor is connected to Bus 1 of MAS and from 121 to 240 if it is connected to Bus 2. These ids are automacally assigned from the system to sensors and they are consecutive considering sensors on the same bus. A special id with value 0 is used is some requests only to give the alarms triggered from MAS and ETH boards with the alarms of all the sensors. A sensor must be in one and only one zone to which propagates its alarms.

This structure describes a sensor

```
{
  "Sensor": {
    "Id": <I:[0-240]>,
    "ZoneId": <I:[1-32]>,
    "ConfigurationId": <I:[1-32]>,
    "IsEnable": <I:[0-1]>,
    "Alarms": <I:bitmask value>,
    "FwVersion": <String:value>,
    "CutCounter": <I:value>,
    "SporadicCutCounter": <I:value>,
  }
}
```

Not all the fields are mandatory, but they could be present depending on the type of request is done. The fields are:

- *Id* is the unique id number of the sensor
- *ZoneId* is the id number of the zone in which the sensor is
- *ConfigurationId* is the id of the configuration from which the sensor take the algorithm parameters
- *IsEnable* indicates if the sensor has to propagate its alarms to zones (and then to outputs)
- *Alarms* is a bitmask with all the alarms of the sensor. See also Alarm value section for a description of alarm values
- *FwVersion* is the fw version of the sensor (available from MAS v1100, ETH v1100, S3H v1010)
- *CutCounter* and *SporadicCutCounter* are the current values of the cut algorithm counters (useful only if the sensor has a mesh configuration type, available from MAS v1100, ETH v1100, S3H v1010)

This structure can be used in a JSON array with key “Sensors”.

3.1.2 Zone data structure

MACS system can group the sensors in at most 32 zone. A zone must be connected to one and only one output to which propagates its alarm.

This structure describes a zone

```
{
  "Zone": {
    "Id": <I:[0-32]>,
    "OutputId": <I:[1-8]>,
    "Name": <String:value>,
    "Alarms": <I:bitmask value>
  }
}
```

Not all the fields are mandatory, but they could be present depending on the type of request is done. The fields are:

- *Id* is the unique id number of the zone
- *OutputId* is the id number of the output connected to the zone
- *Name* is the name of the zone
- *Alarms* is a bitmask with all the alarms of the zone. See also Alarm value section for a description of alarm values

This structure can be repeated in an array with key “Zones”.

3.1.3 Output data structure

MACS system has 8 physical outputs.

This structure describes an output

```
{
  "Output": {
    "Id": <I:[0-8]>,
    "Alarms": <I:bitmask value>
  }
}
```

Not all the fields are mandatory, but they could be present depending on the type of request is done. The fields are:

- *Id* is the unique id number of the output
- *Alarms* is a bitmask with all the alarms of the sensor. See also Alarm value section for a description of alarm values

This structure can be repeated in an array with key “Outputs”.

3.1.4 Configuration data structure

MACS system can handle up to 32 different configurations. A configuration is a set of parameters that can be applied to multiple sensors.

This structure describes a configuration

```
{
  "Configuration": {
    "Id": <I:[1-32]>,
    "Name": <String:value>,
    "CopyFrom": <I:[1-32]>,
    "Type": <I:[1-2]>,
    "Parameters": {
      "Sensibility": <I:value>,
      "Delay": <I:[0-10]>,
      "Axis": <I:[0-2]>,
      "Antiremove": <I:[0-1]>,
      "CalibrationType":<I:[0-1]>,
      "CalibrationId": <I:value>,
      "Calibration": <I:value>,
      "CutSensitivity": <I:[0-2]>,
      "CutDelay": <I:[0-10]>,
      "SporadicCutDelay": <I:[0-10]>,
      "CutReset": <I:[0-65535]>,
      "SporadicCutReset": <I:[0-65535]>,
    }
  }
}
```

Not all the fields are mandatory, but they could be present depending on the type of request is done. The fields are:

- *Id* is the unique id number of the configuration
- *Name* is the name of the configuration
- *CopyFrom* is a value used only during creation of the configuration and requires an Id of a valid configuration to copy its parameters from
- *Type* represents the type of configuration. Its value could be:
  - 1: it is a rigid/semirigid configuration to detect climbing attempts
  - 2: it is a mesh configuration to detect climbing and cut attempts (available from ETH v1100, MAS v1100 and S3H v1010)
- *Parameters* is the set of parameters that describe the configuration made by:
  - *Sensibility* is the value of sensitivity threshold of the climbing algorithm used by the sensor (mandatory for climbing and mesh configurations). Its value could be:
    - From 0 to 10 (up to MAS v1003 and ETH v1006)
    - From 0 to 100 (otherwise)
  - *Delay* is the delay of the climbing algorithm to wait before triggering an alarm (mandatory for climbing and mesh configurations)
  - *Axis* is the axis that the sensor must use (mandatory for climbing and mesh configurations):
    - 0 is X
    - 1 is Y
    - 2 is Z (default and recommended value)
  - *Antiremove* is a value to set on and off anti remove algorithm (mandatory for climbing and mesh configurations):
    - 0 is off
    - 1 is on
  - *CalibrationType* defines the type of fence sensed during calibration (mandatory for climbing and mesh configurations):
    - 0 for rigid fence
    - 1 for semi-rigid fence
  - *CalibrationId* is a id associated to the calibration
  - *Calibration* is a value calculated by the calibration procedure (mandatory for climbing and mesh configurations)
  - *CutSensitivity* is the value of sensitivity threshold of the cut algorithm used by the sensor (mandatory for mesh configurations)

  - *CutDelay* is the delay of the cut algorithm to wait before triggering a cut alarm (mandatory for mesh configurations)
  - *SporadicCutDelay* is the delay of the cut algorithm to wait before triggering a sporadic cut alarm (mandatory for mesh configurations)
  - *CutReset* is the time (in seconds) of the cut algorithm to wait before reset the cut counter
  - *SporadicCutReset* is the time (in minutes) of the cut algorithm to wait before reset the sporadic cut counter

This structure can be repeated in an array with key “Configurations”.

3.1.4.1 Climbing configuration example

```
{
  "Configuration": {
    "Id": 1,
    "Name": "MyConfiguration",
    "Type": 1,
    "Parameters": {
      "Sensibility": 60,
      "Delay": 3,
      "Axis": 2,
      "Antiremove": 1,
      "CalibrationType":1,
      "CalibrationId": 2,
      "Calibration": 3000
    }
  }
}
```

3.1.4.2 Cut configuration example

```
{
  "Configuration": {
    "Id": 1,
    "Name": "MyConfiguration",
    "Type": 2,
    "Parameters": {
      "Sensibility": 60,
      "Delay": 3,
      "Axis": 2,
      "Antiremove": 1,
      "CalibrationType":1,
      "CalibrationId": 2,
      "Calibration": 3000,
      "CutSensitivity": 3,
      "CutDelay": 6,
      "SporadicCutDelay": 6,
      "CutReset": 300,
      "SporadicCutReset": 1440,
    }
  }
}
```

3.1.5 User data structure

MACS system can handle up to 10 different users with different permissions.

This structure describes a user

```
{
  "User": {
    "Id":<I:[1-10]>,
    "Username": <String:value>,
    "Password": <String:value>,
    "OldPassword": <String:value>,
    "ACL": {
      "Write-Zone": <I:[0-1]>,
      "Read-Sensor": <I:[0-1]>,
      "Write-Sensor-Is-Enable": <I:[0-1]>,
      "Write-Sensor-Zone": <I:[0-1]>,
      "Write-Configuration": <I:[0-1]>,
      "Write-Net": <I:[0-1]>,
      "Write-Time": <I:[0-1]>,
      "Fw-Upgrade": <I:[0-1]>,
      "Manage-Users": <I:[0-1]>,
      "Take-System-Control": <I:[0-1]>,
      "Factory-Reset": <I:[0-1]>,
      "Read-Log": <I:[0-1]>,
      "Delete-Log": <I:[0-1]>,
      "Write-Power-Supply": <I:[0-1]>
    },
    "Options": {
      "Change-Password": <I:[0-1]>,
      "Expert-Configuration": <I:[0-1]>
    }
  }
}
```

Not all the fields are mandatory, but they could be present depending on the type of request is done. The fields are:

- *Id* is the unique id number of the user
- *Username* is the username of a user. It can contain letters, numbers or underscores and its lenght must be included from 6 to 16 characters.
- *Password* and *OldPassword* are used only when a user wants to change its own password. Valid passwords have from 6 to 16, must contain only letters or numbers and should have at least a number, a lowercase letter and a uppercase letter
- *ACL* is the list of user's permissions. Every permission can have the value 0 for an action not allowed and 1 for an action allowed. The existing permissions are:
  - *Write-Zone*: if enabled allows to modify/create/delete a zone
  - *Read-Sensor*: if enabled allows to see sensors (data and alarms) and configurations (data)
  - *Write-Sensor-Is-Enable*: if enabled allows to modify *IsEnable* field of sensors
  - *Write-Sensor-Zone*: if enabled allows to modify *ZoneId* field of sensors (so the zone in which is the sensor)
  - *Write-Configuration*: if enabled allows to modify *ConfigurationId* field of sensors (so the parameters used by the sensor), create/delete/modify a configuration, do a calibration on a sensor, read real time data from the sensor
  - *Write-Net*: if enabled allows to modify net parameters

  - *Write-Time*: if enabled allows to modify the date and time values
  - *Fw-Upgrade*: if enabled allows to update the system
  - *Manage-Users*: if enabled allows to modify/create/delete other users
  - *Take-System-Control*: if enabled allows to take system control if someone else is already logged in
  - *Factory-Reset*: if enabled allows to do a factory reset
  - *Read-Log*: if enabled allows to see/download logs
  - *Delete-Log*: if enabled allows to delete logs
  - *Write-Power-Supply*: if enabled allows to enable/disable power supply support
- *Options* indicates operations to be done or special functions
  - *Change-Password* is set to 1 when a password change is required (usually after the first log in)
  - *Expert-Configuration* has a fixed value set to 1 (future purposes)

This structure can be repeated in an array with key “Users”.

3.1.6 Status of operation data structure

This special structure is used in some requests to report the client of the status of a long operation previously required and when the system will be available to a new request.

```
{
  "StatusOfOperation": {
    "Status": <I:value>,
    "SystemStatus": <I:value>,
    "TriggeredVarName": <String: value>,
    "AdditionalData": [<I:value>, <I:value>]
  }
}
```

The fields of the response are:

- *Status* indicates the status of the operation triggered; its value can be:
  - 0: no status available
  - 1: working
  - 2: first phase successful

  - 3: completed successfully
  - 4: aborted/failed
  - 5: working to abort
- *SystemStatus* indicates if the system is working or it's available for the next operation; its value can be:
  - 48: busy with operation
  - 49: working
  - 50: ready/available for new requests
- *TriggeredVarName* has the last triggered command name
- *AdditionalData* are two values used with some command to give some results or some updated information

3.2 Common value description

3.2.1 Alarm value

The alarm value is always specified as a bit mask.

Bit description is the following:

- 0 = Persistent perturbation alarm
- 1 = Persistent perturbation prealarm
- 2 = Climbing alarm
- 3 = Climbing prealarm
- 4 = Remove alarm
- 5 = Magnetic attack alarm
- 6 = Accelerometer fault
- 7 = Accelerometer fault
- 8 = Sensor fault
- 9 = Cut alarm
- 10 = Sporadic cut alarm
- 11 = Cut prealarm
- 12 = Sporadic cut prealarm
- 13 = Not used
- 14 = Not used
- 15 = Not used
- 16 = Communication with sensor failed
- 17 = Sensor not found

- 18 = Bus fault
- 19 = Error writing parameters on sensor
- 20 = Communication security problem
- 21 = Tamper MAS
- 22 = SD card not inserted or damaged
- 23 = Communication MAS-ETH lost
- 24 = Configuration in progress
- 25 = Power supply problems
- 26 = Not used
- 27 = Not used
- 28 = Not used
- 29 = Not used
- 30 = Consistency error
- 31 = Zone or output not use

4 API

The APIs of MACS system are listed below. The title shows the command name and its purpose. In every section there are some useful information like the HTTP methods supported, the URL, the permission required and the previous commands required.

4.1 Login

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: None
- Required previous command execution: None

This API is used for multiple purposes. For example access to MACS and be allowed to make other requests or also logout from the system.

4.1.1 Request

```
{
  "VarName": "Login",
  "Data": {
    "ValueObj": {
      "UserAccess": {
        "Id":<I>,
        "Username":<String:value>,
        "Password":<String:value>,
        "AccessType":<I:[0-4]>
      }
    }
  }
}
```

- *AccessType* is the only mandatory field. Its value must be:
  - 0 = Login (required field “Username” and “Password”)
  - 1 = Logout
  - 2 = Take control
  - 3 = Update JWT Token
  - 4 = Request restore Token (used only during Restore Factory Defaults procedure)
- *Id* field indicates the userId executing the command
- *Username* and *Password* field must be send only with “Login” or “Take Control” requests in clear

4.1.2 Response

The response will be like this:

```
{
  "Result": <I>,
  "Extra": <String:value>
}
```

- *Extra* field contains the JWT token to use in next requests. It will be included only if the request was a “Login”, a “Update JWT Token” or a “Request restore Token” and the request was successful.
- *Result* is a code with the result of the operation. Possible values are:
  - 0 = Successful operation

  - 10 = User not found
  - 11 = Wrong password
  - 12 = System is occupied by another user but it is possible to take control of it
  - 13 = Error

4.1.3 Examples

To do the starting “Login” you have to send

```
{
  "VarName": "Login",
  "Data": {
    "ValueObj": {
      "UserAccess": {
        "Username":<String:value>,
        "Password":<String:value>,
        "AccessType":0
      }
    }
  }
}
```

If the system response is

```
{
  "Result": 0,
  "Extra": <String:value>
}
```

you can use the value of *Extra* field to do next requests.

Otherwise if the response is like that

```
{
  "Result": 12
}
```

you can send another request to the same endpoint like that

```
{
  "VarName": "Login",
  "Data": {
    "ValueObj": {
      "UserAccess": {
        "Username":<String:value>,
        "Password":<String:value>,
        "AccessType":2
      }
    }
  }
}
```

At this point the system response will be like the previous ones to say if the access was granted or not.

4.2 Tool-Connected

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: None
- Required previous command execution: None

This commands allows to set system in a known mode. It's suggested to execute this command after a successful login to set the system in the default monitoring mode because MACS could be in another mode (e.g. when a user take control of the system replacing other user).

4.2.1 Request

```
{
  "VarName": "Tool-Connected",
  "Options": <I:value>
}
```

- *Options* can be one of the following:
  - 1 = Connected
  - 2 = Disconnected
  - 3 = Reset

4.2.2 Response

```
{
  "Result": <I:value>
}
```

4.3 All-Sensors-Data

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: *Read-Sensor*
- Required previous command execution: None

This commands return all the information that describe sensors initializated in MACS in a “Sensors” JSON array.

4.3.1 Request

```
{
  "VarName": "All-Sensors-Data"
}
```

4.3.2 Response

```
{
  "Sensors": [
    {
      "Id": <I:[0-240]>,
      "ZoneId": <I:[1-32]>,
      "ConfigurationId": <I:[1-32]>,
      "IsEnable": <I:[0-1]>,
      "FwVersion": <String:value>
    },
    {
      "Id": <I:[0-240]>,
      "ZoneId": <I:[1-32]>,
      "ConfigurationId": <I:[1-32]>,
      "IsEnable": <I:[0-1]>,
      "FwVersion": <String:value>
    }
  ]
}
```

4.4 All-Zones-Data

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: None
- Required previous command execution: None

This commands return all the information that describe zones created in a “Zones” JSON array.

4.4.1 Request

```
{
  "VarName": "All-Zones-Data"
}
```

4.4.2 Response

```
{
  "Zones": [
    {
      "Id": <I:[0-32]>,
      "OutputId": <I:[1-8]>,
      "Name": <String:value>
    },
    {
      "Id": <I:[0-32]>,
      "OutputId": <I:[1-8]>,
      "Name": <String:value>
    }
  ]
}
```

4.5 All-Configurations-Data

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: *Read-Sensor*
- Required previous command execution: None

This commands return all configurations data in a “Configurations” JSON array.

4.5.1 Request

```
{
  "VarName": "All-Configurations-Data"
}
```

4.5.2 Response

```
{
  "Configurations": [
    {
      "Id": <I:[1-32]>,
      "Name": <String:value>,
      "Type": [1-2],
      "Parameters": {
        "Sensibility": <I:[0-10]>,
        "Delay": <I:[0-10]>,
        "Axis": <I:[0-2]>,
        "Antiremove": <I:[0-1]>,
        "CalibrationType":<I:[0-1]>,
        "CalibrationId": <I:value>,
        "Calibration": <I:value>,
        "CutSensitivity": <I:[0-10]>,
        "CutDelay": <I:[1-10]>,
        "SporadicCutDelay": <I:[0-10]>,
        "CutReset": <I:[0-65535]>,
        "SporadicCutReset": <I:[0-65535]>,
      }
    },
    {
      "Id": <I:[1-32]>,
      "Name": <String:value>,
      "Type": [1-2],
      "Parameters": {
        "Sensibility": <I:[0-10]>,
        "Delay": <I:[0-10]>,
        "Axis": <I:[0-2]>,
        "Antiremove": <I:[0-1]>,
        "CalibrationType":<I:[0-1]>,
        "CalibrationId": <I:value>,
        "Calibration": <I:value>
        "CutSensitivity": <I:[0-10]>,
        "CutDelay": <I:[1-10]>,
        "SporadicCutDelay": <I:[0-10]>,
        "CutReset": <I:[0-65535]>,
        "SporadicCutReset": <I:[0-65535]>,
      }
    }
  ]
}
```

4.6 All-Outputs-Data

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: None
- Required previous command execution: None

This commands return all outputs data in a “Outputs” JSON array (see also

Output data structure section).

4.6.1 Request

```
{
  "VarName": "All-Outputs-Data"
}
```

4.6.2 Response

```
{
  "Outputs": [
    {
      "Id": <I:[0-8]>
    },
    {
      "Id": <I:[0-8]>
    }
  ]
}
```

4.7 All-Sensors-Alarms

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: *Read-Sensor*
- Required previous command execution: None

This commands return all sensors alarms in a "Sensors" JSON array (see also Sensor data structure section).

In the response there is a special sensor with *Id* 0, used to communicate general system alarm (generated from MAS or ETH).

4.7.1 Request

```
{
    "VarName": "All-Sensors-Alarms"
}
```

4.7.2 Response

```
{
  "Sensors": [
    {
      "Id": <I:[0-240]>,
      "Alarms": <I:bitmask value>,
      "IsEnable": <I:[0-1]>,
      "CutCounter": <I:value>,
      "SporadicCutCounter": <I:value>
    },
    {
      "Id": <I:[0-240]>,
      "Alarms": <I:bitmask value>,
      "IsEnable": <I:[0-1]>,
      "CutCounter": <I:value>,
      "SporadicCutCounter": <I:value>
    }
  ]
}
```

4.8 All-Zones-Alarms

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: None
- Required previous command execution: None

This commands return all zones alarms in a “Zones” JSON array (see also Zone data structure section).

In the response there is a special zone with *Id* 0, used to communicate general system alarm (generated from MAS or ETH).

4.8.1 Request

```
{
  "VarName": "All-Zones-Alarms"
}
```

4.8.2 Response

```
{
  "Zones": [
    {
      "Id": <I:[0-32]>,
      "Alarms": <I:bitmask value>
    },
    {
      "Id": <I:[0-32]>,
      "Alarms": <I:bitmask value>
    }
  ]
}
```

4.9 All-Outputs-Alarms

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: None
- Required previous command execution: None

This commands return all outputs alarms in a “Outputs” JSON array (see also Output data structure section).

In the response there is a special output with *Id* 0, used to communicate general system alarm (generated from MAS or ETH).

4.9.1 Request

```
{
  "VarName": "All-Outputs-Data"
}
```

4.9.2 Response

```
{
  "Outputs": [
    {
      "Id": <I:[0-8]>,
      "Alarms": <I:bitmask value>
    },
    {
      "Id": <I:[0-8]>,
      "Alarms": <I:bitmask value>
    }
  ]
}
```

4.10 User-Data

- URL: /generalRequest.json
- HTTP methods: POST/PUT
- Required user permission: None
- Required previous command execution: None

This request can be made to get all users data (depending on permissions) and also to set or change some user data.

4.10.1 Request (POST)

```
{
  "VarName": "User-Data"
}
```

4.10.2 Response (POST)

The response depends on the *Manage-Users* permission of the user that made the request. If this permission is granted the response will contain the complete list of users of the system, otherwise the response will contain only user's own data.

```
{
  "Users": [
    {
      "Id":<I:[1-10]>,
      "Username": <String:value>,
      "ACL": {
        "Write-Zone": <I:[0-1]>,
        "Read-Sensor": <I:[0-1]>,
        "Write-Sensor-Is-Enable": <I:[0-1]>,
        "Write-Sensor-Zone": <I:[0-1]>,
        "Write-Configuration": <I:[0-1]>,
        "Write-Net": <I:[0-1]>,
        "Write-Time": <I:[0-1]>,
        "Fw-Upgrade": <I:[0-1]>,
        "Manage-Users": <I:[0-1]>,
        "Take-System-Control": <I:[0-1]>,
        "Factory-Reset": <I:[0-1]>,
        "Read-Log": <I:[0-1]>,
        "Delete-Log": <I:[0-1]>,
        "Write-Power-Supply": <I:[0-1]>
      },
      "Options": {
        "Change-Password": <I:[0-1]>,
        "Expert-Configuration": 1
      }
    },
    {
      "Id":<I:[1-10]>,
      "Username": <String:value>,
      "ACL": {
        "Write-Zone": <I:[0-1]>,
        "Read-Sensor": <I:[0-1]>,
        "Write-Sensor-Is-Enable": <I:[0-1]>,
        "Write-Sensor-Zone": <I:[0-1]>,
        "Write-Configuration": <I:[0-1]>,
        "Write-Net": <I:[0-1]>,
        "Write-Time": <I:[0-1]>,
        "Fw-Upgrade": <I:[0-1]>,
        "Manage-Users": <I:[0-1]>,
        "Take-System-Control": <I:[0-1]>,
        "Factory-Reset": <I:[0-1]>,
        "Read-Log": <I:[0-1]>,
        "Delete-Log": <I:[0-1]>,
        "Write-Power-Supply": <I:[0-1]>
      },
      "Options": {
        "Change-Password": <I:[0-1]>,
        "Expert-Configuration": <I:[0-1]>
      }
    },
  ]
}
```

4.10.3 Request (PUT)

This request can be made only by users with the *Manage-Users* permission granted. It can be useful to:

- change own password (need only *Password* and *OldPassword*)
- change ACL

The only mandatory value is *Id* which identifies the user on which execute the change.

```
{
  "VarName":"User-Data",
  "Data": {
    "ValueObj": {
      "User": {
        "Id":<I:[1-10]>,
        "Password": <String:value>,
        "OldPassword": <String:value>,
        "ACL": {
          "Write-Zone": <I:[0-1]>,
          "Read-Sensor": <I:[0-1]>,
          "Write-Sensor-Is-Enable": <I:[0-1]>,
          "Write-Sensor-Zone": <I:[0-1]>,
          "Write-Configuration": <I:[0-1]>,
          "Write-Net": <I:[0-1]>,
          "Write-Time": <I:[0-1]>,
          "Fw-Upgrade": <I:[0-1]>,
          "Manage-Users": <I:[0-1]>,
          "Take-System-Control": <I:[0-1]>,
          "Factory-Reset": <I:[0-1]>,
          "Read-Log": <I:[0-1]>,
          "Delete-Log": <I:[0-1]>,
          "Write-Power-Supply": <I:[0-1]>
        },
        "Options": {
          "Change-Password": <I:[0-1]>,
          "Expert-Configuration": <I:[0-1]>
        }
      }
    }
  }
}
```

4.10.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.11 User-Create

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Manage-Users*
- Required previous command execution: None

This request can be made only by users with the *Manage-Users* permission granted. The user created will have the username value as password value. Only *Username* and *ACL* fields are mandatory. All permissions not explicitly indicated will be valued as 0.

4.11.1 Request

```
{
  "VarName":"User-Create",
  "Data": {
    "ValueObj": {
      "User": {
        "Id":<I:[1-10]>,
        "Username": <String:value>,
        "ACL": {
          "Write-Zone": <I:[0-1]>,
          "Read-Sensor": <I:[0-1]>,
          "Write-Sensor-Is-Enable": <I:[0-1]>,
          "Write-Sensor-Zone": <I:[0-1]>,
          "Write-Configuration": <I:[0-1]>,
          "Write-Net": <I:[0-1]>,
          "Write-Time": <I:[0-1]>,
          "Fw-Upgrade": <I:[0-1]>,
          "Manage-Users": <I:[0-1]>,
          "Take-System-Control": <I:[0-1]>,
          "Factory-Reset": <I:[0-1]>,
          "Read-Log": <I:[0-1]>,
          "Delete-Log": <I:[0-1]>,
          "Write-Power-Supply": <I:[0-1]>
        }
      }
    }
  }
}
```

4.11.2 Response

```
{
  "Result": <I:value>
}
```

4.12 User-Reset

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Manage-Users*
- Required previous command execution: None

This request can be made only by users with the *Manage-Users* permission granted. The user resetted will have the same ACL but the password will be set to username value.

4.12.1 Request

```
{
  "VarName":"User-Reset",
  "Data": {
    "ValueObj": {
      "User": {
        "Id": <I:[1-10]>
      }
    }
  }
}
```

4.12.2 Response

```
{
  "Result": <I:value>
}
```

4.13 User-Delete

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Manage-Users*
- Required previous command execution: None

This request can be made only by users with the *Manage-Users* permission granted. It's not possible to delete the administrator user (named "admin").

4.13.1 Request

```
{
  "VarName":"User-Delete",
  "Data": {
    "ValueObj": {
      "User": {
        "Id": <I:[1-10]>
      }
    }
  }
}
```

4.13.2 Response

```
{
  "Result": <I:value>
}
```

4.14 Zone-Create

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Write-Zone*
- Required previous command execution: None

This request can be made only by users with the *Write-Zone* permission granted. The system creates a zone and assigns automatically a new id to the zone. The new zone must be in a existing zone and can't have the same name of another zone.

4.14.1 Request

```
{
  "VarName": "Zone-Create",
  "Data": {
    "ValueObj": {
      "Zone": {
        "OutputId": <I:[1-8]>,
        "Name": <String:value>
      }
    }
  }
}
```

4.14.2 Response

```
{
  "Result": <I:value>
}
```

4.15 Zone-Delete

- URL: /generalRequest.json
- HTTP methods: PUT

- Required user permission: *Write-Zone*
- Required previous command execution: None

This request can be made only by users with the *Write-Zone* permission granted. The zone to be deleted must exist and must be empty (without sensor inside it).

4.15.1 Request

```
{
  "VarName": "Zone-Delete",
  "Data": {
    "ValueObj": {
      "Zone": {
        "Id": <I:[1-32]>
      }
    }
  }
}
```

4.15.2 Response

```
{
  "Result": <I:value>
}
```

4.16 Zone-OutputId

- URL: /generalRequest.json
- HTTP methods: POST/PUT
- Required user permission: *Write-Zone* (PUT) or None (POST)
- Required previous command execution: None

This request can be submitted with POST (read) or PUT (write) HTTP method. The POST request can be made by any user but the PUT requests can be made only by users with the *Write-Zone* permission granted.

4.16.1 Request (POST)

```
{
  "VarName": "Zone-OutputId",
  "Data": {
    "ValueObj": {
      "Zone": {
        "Id": <I:[1-32]>
      }
    }
  }
}
```

4.16.2 Response (POST)

```
{
  "Zones": [
    {
      "Id": <I:[1-32]>,
      "OutputId": <I:[1-8]>
    }
  ]
}
```

4.16.3 Request (PUT)

```
{
  "VarName": "Zone-OutputId",
  "Data": {
    "ValueObj": {
      "Zone": {
        "Id": <I:[1-32]>,
        "OutputId": <I:[1-8]>
      }
    }
  }
}
```

4.16.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.17 Zone-Name

- URL: /generalRequest.json
- HTTP methods: POST/PUT
- Required user permission: *Write-Zone* (PUT) or None (POST)
- Required previous command execution: None

This request can be submitted with POST (read) or PUT (write) HTTP method. The POST request can be made by any user but the PUT requests can be made only by users with the *Write-Zone* permission granted. When it's changed the name of a zone, it cannot be the same name of another zone.

4.17.1 Request (POST)

```
{
  "VarName": "Zone-Name",
  "Data": {
    "ValueObj": {
      "Zone": {
        "Id": <I:[1-32]>
      }
    }
  }
}
```

4.17.2 Response (POST)

```
{
  "Zones": [
    {
      "Id": <I:[1-32]>,
      "Name": <String:value>
    }
  ]
}
```

4.17.3 Request (PUT)

```
{
  "VarName": "Zone-Name",
  "Data": {
    "ValueObj": {
      "Zone": {
        "Id": <I:[1-32]>,
        "Name": <String:value>
      }
    }
  }
}
```

4.17.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.18 Zone-Alarms

- URL: /generalRequest.json
- HTTP methods: POST

- Required user permission: None
- Required previous command execution: None

This request can be made by any user and is useful to retrieve the alarms of a single zone.

4.18.1 Request

```
{
  "VarName": "Zone-Alarms",
  "Data": {
    "ValueObj": {
      "Zone": {
        "Id": <I:[1-32]>
      }
    }
  }
}
```

4.18.2 Response

```
{
  "Zones": [
    {
      "Id": <I:[1-32]>,
      "Alarms": <I:bitmask value>
    }
  ]
}
```

4.19 Sensor-ZoneId

- URL: /generalRequest.json
- HTTP methods: POST/PUT
- Required user permission: *Read-Sensor* (POST) or *Write-Sensor-Zone* (PUT)

- Required previous command execution: None

This request can be submitted with POST (read) or PUT (write) HTTP method. The POST request can be made by any user with *Read-Sensor* permission. The PUT requests can be made only by users with the *Write-Sensor-Zone* permission granted.

4.19.1 Request (POST)

```
{
  "VarName": "Sensor-ZoneId",
  "Data": {
    "ValueObj": {
      "Sensor": {
        "Id": <I:[1-240]>
      }
    }
  }
}
```

4.19.2 Response (POST)

```
{
  "Zones": [
    {
      "Id": <I:[1-240]>,
      "ZoneId": <I:[1-32]>
    }
  ]
}
```

4.19.3 Request (PUT - 1)

This request is used to change the zone of a single sensor.

```
{
  "VarName": "Sensor-ZoneId",
  "Data": {
    "ValueObj": {
      "Sensor": {
        "Id": <I:[1-240]>,
        "ZoneId": <I:[1-32]>
      }
    }
  }
}
```

4.19.4 Response (PUT - 1)

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.19.5 Request (PUT - 2)

This request can be used to assign the same zone to one or more sensor.

```
{
  "VarName": "Sensor-ZoneId",
  "Data": {
    "Value": <I:[1-32]>,
    "Sensors": [
      {
        "Id": <I:[1-240]>
      },
      {
        "Id": <I:[1-240]>
      }
    ]
  }
}
```

4.19.6 Response (PUT - 2)

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.20 Sensor-ConfigurationId

- URL: /generalRequest.json
- HTTP methods: POST/PUT
- Required user permission: *Read-Sensor* (POST) or *Write-Configuration* (PUT)
- Required previous command execution: None

This request can be submitted with POST (read) or PUT (write) HTTP method. The POST request can be made by any user with *Read-Sensor* permission. The PUT requests can be made only by users with the *Write-Configuration* permission granted.

4.20.1 Request (POST)

```
{
  "VarName": "Sensor-ConfigurationId",
  "Data": {
    "ValueObj": {
      "Sensor": {
        "Id": <I:[1-240]>
      }
    }
  }
}
```

4.20.2 Response (POST)

```
{
  "Zones": [
    {
      "Id": <I:[1-240]>,
      "ConfigurationId": <I:[1-32]>
    }
  ]
}
```

4.20.3 Request (PUT - 1)

This request is used to change the configuration of a single sensor.

```
{
  "VarName": "Sensor-ConfigurationId",
  "Data": {
    "ValueObj": {
      "Sensor": {
        "Id": <I:[1-240]>,
        "ConfigurationId": <I:[1-32]>
      }
    }
  }
}
```

4.20.4 Response (PUT - 1)

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.20.5 Request (PUT - 2)

This request can be used to assign the same configuration to one or more sensor.

```
{
  "VarName": "Sensor-ConfigurationId",
  "Data": {
    "Value": <I:[1-32]>,
    "Sensors": [
      {
        "Id": <I:[1-240]>
      },
      {
        "Id": <I:[1-240]>
      }
    ]
  }
}
```

4.20.6 Response (PUT - 2)

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.21 Sensor-Is-Enable

- URL: /generalRequest.json
- HTTP methods: POST/PUT
- Required user permission: *Read-Sensor* (POST) or *Write-Sensor-Is-Enable* (PUT)
- Required previous command execution: None

This request can be submitted with POST (read) or PUT (write) HTTP method. The POST request can be made by any user with *Read-Sensor* permission. The PUT requests can be made only by users with the *Write-Sensor-Is-Enable* permission granted.

4.21.1 Request (POST)

```
{
  "VarName": "Sensor-Is-Enable",
  "Data": {
    "ValueObj": {
      "Sensor": {
        "Id": <I:[1-240]>
      }
    }
  }
}
```

4.21.2 Response (POST)

```
{
  "Sensors": [
    {
      "Id": <I:[1-240]>,
      "IsEnable": <I:[0-1]>
    }
  ]
}
```

4.21.3 Request (PUT - 1)

This request is used to change the state of a single sensor.

```
{
  "VarName": "Sensor-Is-Enable",
  "Data": {
    "ValueObj": {
      "Sensor": {
        "Id": <I:[1-240]>,
        "IsEnable": <I:[0-1]>
      }
    }
  }
}
```

4.21.4 Response (PUT - 1)

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.21.5 Request (PUT - 2)

This request can be used to assign the same state to one or more sensor.

```
{
  "VarName": "Sensor-Is-Enable",
  "Data": {
    "Value": <I:[0-1]>,
    "Sensors": [
      {
        "Id": <I:[1-240]>
      },
      {
        "Id": <I:[1-240]>
      }
    ]
  }
}
```

4.21.6 Response (PUT - 2)

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.22 Sensor-Alarms

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: *Read-Sensor* (POST)
- Required previous command execution: None

This request can be made by any user that has the *Read-Sensor* permission and is useful to retrieve the alarms of a single sensor.

4.22.1 Request

```
{
  "VarName": "Sensor-Alarms",
  "Data": {
    "ValueObj": {
      "Sensor": {
        "Id": <I:[1-240]>
      }
    }
  }
}
```

4.22.2 Response

```
{
  "Sensors": [
    {
      "Id": <I:[1-240]>,
      "Alarms": <I:bitmask value>
      "CutCounter": <I:value>,
      "SporadicCutCounter": <I:value>
    }
  ]
}
```

4.23 Sensor-Counter

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Write-Configuration* (PUT)
- Required previous command execution: None

This request can be submitted with PUT (write) HTTP method. The PUT requests can be made only by users with the *Write-Configuration* permission granted.

4.23.1 Request (PUT)

This request can be used to assign the same state to one or more sensor.

```
{
  "VarName": "Sensor-Counter",
  "Data": {
    "Value": <I:value>,
    "Sensors": [
      {
        "Id": <I:[1-240]>
      },
      {
        "Id": <I:[1-240]>
      }
    ]
  }
}
```

4.23.2 Response

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.24 Output-Alarms

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: None
- Required previous command execution: None

This request can be made by all users.

4.24.1 Request

```
{
  "VarName": "Output-Alarms",
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-8]>
      }
    }
  }
}
```

4.24.2 Response

```
{
  "Outputs": [
    {
      "Id": <I:[1-8]>,
      "Alarms": <I:bitmask value>
    },
    {
      "Id": <I:[1-8]>,
      "Alarms": <I:bitmask value>
    }
  ]
}
```

4.25 Configuration-Name

- URL: /generalRequest.json
- HTTP methods: POST/PUT
- Required user permission: *Read-Sensor* (POST) or *Write-Configuration* (PUT)
- Required previous command execution: None

This request can be submitted with POST (read) or PUT (write) HTTP method. The POST request can be made by any user with *Read-Sensor* permission. The PUT requests can be made only by users with the *Write-Configuration* permission granted.

4.25.1 Request (POST)

```
{
  "VarName": "Configuration-Name",
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>
      }
    }
  }
}
```

4.25.2 Response (POST)

```
{
  "Configurations": [
    {
      "Id": <I:[1-32]>,
      "Name": <String:value>
    }
  ]
}
```

4.25.3 Request (PUT)

When it's changed the name of a configuration, duplicates are not allowed.

```
{
  "VarName": "Configuration-Name",
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>,
        "Name": <String:value>
      }
    }
  }
}
```

4.25.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.26 Configuration-Delete

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Write-Configuration* (PUT)
- Required previous command execution: None

This request can be made only by users who has the *Write-Configuration* permission. It's possible to delete a configuration only if it is associated with no sensors.

4.26.1 Request

```
{
  "VarName": "Configuration-Delete",
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>
      }
    }
  }
}
```

4.26.2 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.27 Configuration-Parameters

- URL: /generalRequest.json
- HTTP methods: POST/PUT
- Required user permission: *Read-Sensor* (POST) or *Write-Configuration* (PUT)
- Required previous command execution: Sensor-Configuration-Mode (PUT) or None (POST)

This request can be submitted with POST (read) or PUT (write) HTTP method. The POST request can be made by any user with *Read-Sensor* permission. The PUT requests can be made only by users with the *Write-Configuration* permission granted.

4.27.1 Request (POST)

```
{
  "VarName": "Configuration-Parameters",
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>
      }
    }
  }
}
```

4.27.2 Response (POST)

```
{
  "Configurations": [
    {
      "Id": <I:[1-32]>,
      "Type": [1-2],
      "Parameters": {
        "Sensibility": <I:[0-100]>,
        "Delay": <I:[0-10]>,
        "Axis": <I:[0-2]>,
        "Antiremove": <I:[0-1]>,
        "CalibrationType":<I:[0-1]>,
        "CalibrationId": <I:value>,
        "Calibration": <I:value>,
        "CutSensitivity": <I:[0-10]>,
        "CutDelay": <I:[1-10]>,
        "SporadicCutDelay": <I:[0-10]>,
        "CutReset": <I:[0-65535]>,
        "SporadicCutReset": <I:[0-65535]>,
      }
    }
  ]
}
```

CutSensitivity, *CutDelay*, *SporadicCutDelay*, *CutReset* and *SporadicCutReset* fields will be included also if the configuration *Type* field is 2 (Mesh configuration).

4.27.3 Request (PUT)

```
{
  "VarName": "Configuration-Parameters",
  "Options": <I:[1-3]>,
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>
        "Type": [1-2],
        "Parameters": {
          "Sensibility": <I:value>,
          "Delay": <I:[0-10]>,
          "Axis": <I:[0-2]>,
          "Antiremove": <I:[0-1]>,
          "CalibrationType":<I:[0-1]>,
          "CalibrationId": <I:value>,
          "Calibration": <I:value>,
          "CutSensitivity": <I:[0-10]>,
          "CutDelay": <I:[1-10]>,
          "SporadicCutDelay": <I:[0-10]>,
          "CutReset": <I:[0-65535]>,
          "SporadicCutReset": <I:[0-65535]>
        }
      }
    }
  }
}
```

The type of operation requested change depending on *Options* value like this:

- 1: it's requested a change of parameters only for test purposes; in this case the modifies are temporary and not saved
- 2: it's requested a change of parameters to save new parameters; after that the new parameters are written to all sensors with this configuration assigned
- 3: it's requested to reset the parameters to starting values (probably after a test)

To change the configuration parameters it's possible:

- specify some new values in *Parameters* object like *Sensibility*, *Delay*, *Axis*, *Antiremove*, *CutSensitivity*, *CutDelay*, *SporadicCutDelay*, *CutReset* and *SporadicCutReset* fields (recommended)
- specify a different *Type* of configuration

```
{
  "VarName": "Configuration-Parameters",
  "Options": <I:[1-3]>,
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>
        "Type": [1-2],
        "Parameters": {
          "Sensibility": <I:value>,
          "Delay": <I:[0-10]>,
          "Axis": <I:[0-2]>,
          "Antiremove": <I:[0-1]>
          "CutSensitivity": <I:[0-10]>,
          "CutDelay": <I:[1-10]>,
          "SporadicCutDelay": <I:[0-10]>,
          "CutReset": <I:[0-65535]>,
          "SporadicCutReset": <I:[0-65535]>,
        }
      }
    }
  }
}
```

- All the fields inside *Parameters*

```
{
  "VarName": "Configuration-Parameters",
  "Options": <I:[1-3]>,
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>
        "Type": 1,
        "Parameters": {
          "Sensibility": <I:value>,
          "Delay": <I:[0-10]>,
          "Axis": <I:[0-2]>,
          "Antiremove": <I:[0-1]>,
          "CalibrationType":<I:[0-1]>,
          "CutSensitivity": <I:[0-10]>,
          "CutDelay": <I:[1-10]>,
          "SporadicCutDelay": <I:[0-10]>,
          "CutReset": <I:[0-65535]>,
          "SporadicCutReset": <I:[0-65535]>,
        }
      }
    }
  }
}
```

4.27.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.28 Configuration-Create

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Write-Configuration*
- Required previous command execution: Sensor-Configuration-Mode

This request can be made only by users with the *Write-Configuration* permission granted. This request is useful to create a new configuration and directly associate to the sensor under test.

4.28.1 Request

```
{
  "VarName": "Configuration-Create",
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>,
        "Name": <String:value>,
        "CopyFrom": <I:[1-32]>,
        "Type": [1-2],
        "Parameters": {
          "Sensibility": <I:value>,
          "Delay": <I:[0-10]>,
          "Axis": <I:[0-2]>,
          "Antiremove": <I:[0-1]>,
          "CalibrationType":<I:[0-1]>,
          "CalibrationId": <I:value>,
          "Calibration": <I:value>,
          "CutSensitivity": <I:[0-10]>,
          "CutDelay": <I:[1-10]>,
          "SporadicCutDelay": <I:[0-10]>,
          "CutReset": <I:[0-65535]>,
          "SporadicCutReset": <I:[0-65535]>,
        }
      }
    }
  }
}
```

There are different ways to create a configuration and assign the calibration data to it, specifying:

- *CopyFrom* field with the id of a configuration from which calibration data are copied (recommended)

```
{
  "VarName": "Configuration-Create",
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>,
        "Name": <String:value>,
        "CopyFrom": <I:[1-32]>,
        "Type": [1-2],
        "Parameters": {
          "Sensibility": <I:value>,
          "Delay": <I:[0-10]>,
          "Axis": <I:[0-2]>,
          "Antiremove": <I:[0-1]>
          "CutSensitivity": <I:[0-10]>,
          "CutDelay": <I:[1-10]>,
          "SporadicCutDelay": <I:[0-10]>,
          "CutReset": <I:[0-65535]>,
          "SporadicCutReset": <I:[0-65535]>,
        }
      }
    }
  }
}
```

- *CalibrationType*, *CalibrationId* and *Calibration* inside the *Parameters* object

```
{
  "VarName": "Configuration-Create",
  "Data": {
    "ValueObj": {
      "Configuration": {
        "Id": <I:[1-32]>,
        "Name": <String:value>,
        "Type": [1-2],
        "Parameters": {
          "Sensibility": <I:value>,
          "Delay": <I:[0-10]>,
          "Axis": <I:[0-2]>,
          "Antiremove": <I:[0-1]>,
          "CalibrationType":<I:[0-1]>,
          "CalibrationId": <I:value>,
          "Calibration": <I:value>,
          "CutSensitivity": <I:[0-10]>,
          "CutDelay": <I:[1-10]>,
          "SporadicCutDelay": <I:[0-10]>,
          "CutReset": <I:[0-65535]>,
          "SporadicCutReset": <I:[0-65535]>,
        }
      }
    }
  }
}
```

4.28.2 Response

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.29 Sensor-Configuration-Mode

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Write-Configuration* (PUT)
- Required previous command execution: None

This request can be made only by users with the *Write-Configuration* permission granted. This request is mandatory to modify/create/test a configuration and it's useful for the system to identify the sensor used as test during this operations.

4.29.1 Request

```
{
  "VarName": "Sensor-Configuration-Mode",
  "Data": {
    "ValueObj": {
      "Sensor": {
        "Id": <I:[1-240]>
      }
    }
  }
}
```

4.29.2 Response

```
{
  "Result": <I:value>
}
```

4.30 Last-Operation-Status

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: None
- Required previous command execution: None

This request can be made by any user. When a successful PUT request (of any type) require some long operations and it's result value is 100, it's mandatory for the client to start asking this request to know when the system finished the operation and it is available to fulfill other requests again.

4.30.1 Request

```
{
  "VarName": "Last-Operation-Status"
}
```

4.30.2 Response

```
{
  "StatusOfOperation": {
    "Status": <I:value>,
    "SystemStatus": <I:value>,
    "TriggeredVarName": <String: value>,
    "AdditionalData": [<I:value>, <I:value>]
  }
}
```

4.31 Init-Sensors

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Write-Configuration*
- Required previous command execution: None

This request can be made by any user with the *Write-Configuration* permission. It is useful to initialize the sensor in the system and assign automatically to them an unique id.

4.31.1 Request

```
{
  "VarName": "Init-Sensors",
  "Options": <I:value>
}
```

This request triggers different initialization operations, depending on *Options* value:

- 0 - complete initialization: the initialization procedure resets all the old sensor data (like zone and configuration assigned to them); the system assign to the sensor some default data
- 1 - partial initialization: the initialization procedure tries to keep all the current sensor data; the system will set the same data on all the

old sensor found, will set the default data for new one and delete all the informations of sensors not found

Pay attencion on the partial initialization because the id assigned from the system to the sensors are always sequential, so a sensor could not have the same id between two initializations, e.g.:

- Connect and initialize 3 sensors A, B, C. The situation will be like that A (id = 1), B (id = 2), C (id = 3)
- A new sensor D is added between A and B
- After executing a partial initialization, the situation will be like that A (id = 1), D (id = 2), B (id = 3), C (id = 4). So the information of sensor B (old sensor with id 2) will be set to sensor D (new sensor with id 2) and so on

4.31.2 Response

```
{
  "Result": <I:value>
}
```

The *Result* value will be different from 100 if the request failed. Otherwise (as the result value asks) it is requested to the client to ask some *Last-Operation-Status* request. If the operation finishes successfully, the number of sensors found will be in the *AdditionalData* field (the first value is the number of sensor on BUS 1, the other one on BUS 2).

4.32 Search-Sensor

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: None
- Required previous command execution: None

This request can be made by any user. It is useful to trigger the procedure to search/identify a sensor with a magnet.

4.32.1 Request

```
{
  "VarName": "Search-Sensor",
  "Options": <I:value>
}
```

This request triggers different state of identification operation, depending on *Options* value:

- 1 - start identification: start the identification procedure
- 2 - abort identification: abort a previously started identification procedure

4.32.2 Response

```
{
  "Result": <I:value>
}
```

The *Result* value will be different from 100 if the request failed. Otherwise (as the result value asks) it is requested to the client to ask some *Last-Operation-Status* request. When a sensor is found, the operation is automatically finished and the id of the sensor found will be in the *Additionaldata* field.

4.33 Calibration-Mask-Mode

- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Write-Configuration*
- Required previous command execution: Sensor-Configuration-Mode

This request can be made by any user with *Write-Configuration* permission. It is useful to trigger the procedure to calculate the calibration for a configuration on the sensor under test.

The system can require one or two phases to complete this procedure successfully.

4.33.1 Request

```
{
  "VarName": "Calibration-Mask-Mode",
  "Options": <I:value>
}
```

This request triggers different state of calibration calculation, depending on *Options* value:

- 1 - start calibration: start the calibration procedure
- 2 - abort calibration: abort a previously started calibration procedure

4.33.2 Response

```
{
  "Result": <I:value>
}
```

The *Result* value will be different from 100 if the request failed. Otherwise (as the result value asks) it is requested to the client to ask some *Last-Operation-Status* request. *Status* field of *Last-Operation-Status* request indicates if the operation is working, aborted, finished or has phase one successful. In this last case the second phase will be automatically triggered by the system. When the operation finished and the system returns available to new requests (it can be seen from *SystemStatus* field of *Last-Operation-Status*), the system returns through the *AdditionalData* of *Last-Operation-Status* response respectively the type of fence (as in the *IsResonant* field of *Configuration*) and the new calibrationId (as in the *CalibrationId* field of *Configuration*).

4.34 Sensor-Test-Data

- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: *Write-Configuration*
- Required previous command execution: Sensor-Configuration-Mode

This request can be made by any user with *Write-Configuration* permission. It is useful to ask some current data on the sensor under test.

4.34.1 Request

```
{
  "VarName": "Sensor-Test-Data"
}
```

4.34.2 Response (Up to MAS 1100 and ETH 1100)

```
{
  "TestData": {
    "Alarms": <I:bitmask value>,
    "CurrentEnergy": <I:value>,
    "CurrentEnergyThreshold": <I:value>,
    "CurrentAlarmCounter": <I:value>,
    "CurrentResetCounter": <I:value>,
    "CurrentDelay": <I:value>
  }
}
```

The fields indicates:

- *Alarms* is a bitmask value with the current alarms triggered on the sensor
- *CurrentEnergy* is the current energy sensed by the sensor
- *CurrentEnergyThreshold* is the current threshold on the energy value used by the sensor
- *CurrentAlarmCounter* is the current delay value on the sensor
- *CurrentResetCounter* is the current reset value
- *CurrentDelay* is the current value of delay set on the sensor

4.34.3 Response

```
{
  "TestData": {
    "Alarms": <I:bitmask value>,
    "Type": [1-2],
    "Data": {
      "CurrentEnergy": <I:value>,
      "CurrentEnergyThreshold": <I:value>,
      "CurrentAlarmCounter": <I:value>,
      "CurrentResetCounter": <I:value>,
      "CurrentDelay": <I:value>,
      "CurrentCutCounter": <I:value>,
      "CurrentSporadicCutCounter": <I:value>,
      "CurrentCutEnergy": <I:value>,
      "CurrentCutSensitivity": <I:value>,
      "CurrentCutAlarm": <I:value>,
      "CurrentSporadicCutAlarm": <I:value>,
      "CurrentCutSensitivityThreshold": <I:value>
    }
  }
}
```

The fields indicates:

- *Alarms* is a bitmask value with the current alarms triggered on the sensor
- *Type* is the current configuration type set
- *Data* contains the current data:
  - *CurrentEnergy* is the current climbing energy sensed by the sensor
  - *CurrentEnergyThreshold* is the current climbing threshold on the energy value used by the sensor
  - *CurrentAlarmCounter* is the current climbing delay value on the sensor
  - *CurrentResetCounter* is the current climbing reset value
  - *CurrentDelay* is the current value of climbing delay set on the sensor
  - *CurrentCutCounter* is the current value of cut counter on the sensor
  - *CurrentSporadicCutCounter* is the current value of sporadic cut counter on the sensor
  - *CurrentCutEnergy* is the current cut energy sensed by the sensor
  - *CurrentCutSensitivity* is the current cut sensitivity sensed by the

  sensor
  - *CurrentCutAlarm* is the current threshold on the cut counter used by the sensor
  - *CurrentSporadicCutAlarm* is the current threshold on the sporadic cut counter used by the sensor
  - *CurrentCutSensitivityThreshold* is the current threshold on the sensitivity used by the sensor

4.35 Net-Configuration

- URL: /generalRequest.json
- HTTP methods: PUT/POST
- Required user permission: *Write-Net* (PUT) or None (POST)
- Required previous command execution: None

This request in POST method can be made by all the users but the ones with PUT method can be made only by the users with *Write-Net* permission. It is useful to get/set the mode on the net.

4.35.1 Request (POST)

```
{
  "VarName": "Net-Configuration"
}
```

4.35.2 Response (POST)

```
{
  "Value": <I:bitmask value>
}
```

The *Value* field, that represents the current configuration, is a bitmask value composed like this:

- bit 0: DHCP (set to 1)/static IP (set to 0)
- bit 1: name resolution (through NetBIOS or mDNS)

4.35.3 Request (PUT)

```
{
  "VarName": "Net-Configuration",
  "Data": {
    "Value": <I:bitmask value>
  }
}
```

Where *Value* field is specified like the description in the previous section.

4.35.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.36 IPV4-Address

- URL: /generalRequest.json
- HTTP methods: PUT/POST
- Required user permission: *Write-Net* (PUT) or None (POST)
- Required previous command execution: None

This request in POST method can be made by all the users but the ones with PUT method can be made only by the users with *Write-Net* permission. It is useful to get/set the static ip configurations. The use of these configurations depends on the value set in *Net-Configuration* property.

4.36.1 Request (POST)

```
{
  "VarName": "IPV4-Address"
}
```

4.36.2 Response (POST)

```
{
  "StaticIPV4Config": {
    "Address": <String:value>,
    "Subnet": <String:value>,
    "Gateway": <String:value>,
    "PrimaryDNS": <String:value>,
    "SecondaryDNS": <String:value>
  }
}
```

The fields in *StaticIPV4Config* are:

- *Address* is the static address of the system (in a string value as "xxx.xxx.xxx.xxx")
- *Subnet* is the subnet used by the system
- *Gateway* is the gateway used by the system
- *PrimaryDNS* is the primary DNS address pointed from the system
- *SecondaryDNS* is the secondary DNS address pointed from the system

4.36.3 Request (PUT)

```
{
  "VarName": "IPV4-Address",
  "Data": {
    "ValueObj": {
      "StaticIPV4Config": {
        "Address": <String:value>,
        "Subnet": <String:value>,
        "Gateway": <String:value>,
        "PrimaryDNS": <String:value>,
        "SecondaryDNS": <String:value>
      }
    }
  }
}
```

The fields in the *StaticIPV4Config* are specified like the description in the previous section.

4.36.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.37 Time

- URL: /generalRequest.json
- HTTP methods: PUT/POST
- Required user permission: *Write-Time* (PUT) or None (POST)
- Required previous command execution: None

This request in POST method can be made by all the users but the ones with PUT method can be made only by the users with *Write-Time* permission. It is useful to get/set the date and time values.

4.37.1 Request (POST)

```
{
  "VarName": "Time"
}
```

4.37.2 Response (POST)

```
{
  "DateTime": {
    "Time": <I:value>,
    "UTC": <I:value>
  }
}
```

The *DateTime* object is composed by:

- *Time* is the value in unix time referred to time zone +00:00
- *UTC* is the value to indicate the timezone in seconds (e.g. 3600 = +01:00, 20700 = +05:45)

4.37.3 Request (PUT)

```
{
  "VarName": "Time",
  "Data": {
    "ValueObj": {
      "DateTime": {
        "Time": <I:value>,
        "UTC": <I:value>
      }
    }
  }
}
```

Where the fields in the *DateTime* object are like the ones in the previous section.

4.37.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.38 Eth-Restore-Factory-Defaults

- URL: /restoreFactoryReset.json
- HTTP methods: PUT
- Required user permission: *Factory-Reset*
- Required previous command execution: None

This request can be made by any user with *Factory-Reset* permission. This request is useful to reset to factory defaults settings about users, net parameters, time and power supply.

4.38.1 Request

```
{
  "VarName": "Eth-Restore-Factory-Defaults"
}
```

4.38.2 Response

```
{
  "Result": <I:value>
}
```

4.39 Master-Restore-Factory-Defaults

- URL: /restoreFactoryReset.json
- HTTP methods: PUT
- Required user permission: *Factory-Reset*
- Required previous command execution: None

This request can be made by any user with *Factory-Reset* permission. This request is useful to reset to factory defaults settings about sensors, zones, configurations and outputs.

4.39.1 Request

```
{
  "VarName": "Master-Restore-Factory-Defaults"
}
```

4.39.2 Response

```
{
  "Result": <I:value>
}
```

The response *Result* will be 100 if the request is successful. With this value is requested to the client to continuously request Last-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

4.40 Power-Supply-Enable

- URL: /generalRequest.json
- HTTP methods: PUT/POST
- Required user permission: *Write-Power-Supply* (PUT) or None (POST)
- Required previous command execution: None

This request with POST method can be made by all the users but the ones with PUT method any user with *Write-Power-Supply* permission. It is useful to enable/disable the check of power supply.

4.40.1 Request (POST)

```
{
  "VarName": "Power-Supply-Enable"
}
```

4.40.2 Response (POST)

```
{
  "Value": <I:[0-1]>
}
```

The field *Value* indicates if the check of power supply status is enabled (1) or not (0).

4.40.3 Request (PUT)

```
{
  "VarName": "Power-Supply-Enable",
  "Data": {
    "Value": <I:[0-1]>
  }
}
```

Where *Value* field is specified in the description in the previous section.

4.40.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.41 Power-Supply-Status

- URL: /generalRequest.json
- HTTP methods: PUT/POST
- Required user permission: *Write-Power-Supply* (PUT) or None (POST)
- Required previous command execution: None

This request with POST method can be made by all the users but the ones with PUT method any user with *Write-Power-Supply* permission. It is useful to check if the status of power supply.

4.41.1 Request (POST)

```
{
  "VarName": "Power-Supply-Enable"
}
```

4.41.2 Response (POST)

```
{
  "Value": <I:bitmask value>
}
```

The field *Value* indicates if the status of power supply in a bitmask composed by:

- bit 0: Power loss
- bit 1: Power problem
- bit 2: Battery low
- bit 3: Tamper power supply

4.41.3 Request (PUT)

```
{
  "VarName": "Power-Supply-Enable",
  "Data": {
    "Value": <I:bitmask value>
  }
}
```

Where *Value* field is specified in the description in the previous section.

4.41.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.42 Master-Tamper-Enabled

- URL: /generalRequest.json
- HTTP methods: PUT/POST
- Required user permission: *Write-Power-Supply* (PUT) or None (POST)
- Required previous command execution: None

This request with POST method can be made by all the users but the ones with PUT method any user with *Write-Power-Supply* permission. It is useful to check if the status of power supply.

4.42.1 Request (POST)

```
{
  "VarName": "Master-Tamper-Enabled"
}
```

4.42.2 Response (POST)

```
{
  "Value": <I:[0-1]>
}
```

The field *Value* indicates if the check of tamper is enabled or disabled on MAS board.

4.42.3 Request (PUT)

```
{
  "VarName": "Master-Tamper-Enabled",
  "Data": {
    "Value": <I:[0-1]>
  }
}
```

Where *Value* field is specified in the description in the previous section.

4.42.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

4.43 Get-Device-Info

- URL: /status.json
- HTTP methods: POST
- Required user permission: None
- Required previous command execution: None

This request can be made by all the users.

4.43.1 Request

```
{
  "VarName": "Get-Device-Info"
}
```

4.43.2 Response

```
{
  "DeviceInfo": {
    "EthHwVersion": <String:value>,
    "EthFwVersion": <String:value>,
    "EthSerialId": <String:value>,
    "MasterHwVersion": <String:value>,
    "MasterFwVersion": <String:value>,
    "MasterSerialId": <String:value>,
    "SensorMaxHwVersion": <String:value>,
    "SensorMinHwVersion": <String:value>,
    "SensorMaxFwVersion": <String:value>,
    "SensorMinFwVersion": <String:value>
  }
}
```

The field *DeviceInfo* is composed by:

- *EthHWVersion* is the HW version of ETH board
- *EthFWVersion* is the FW version of ETH board
- *EthSerialId* is the serial of ETH board
- *MasterHWVersion* is the HW version of MAS board

- *MasterFWVersion* is the FW version of MAS board
- *MasterSerialId* is the serial of MAS board
- *SensorMaxHwVersion* is the max HW version of the sensors initializated in the system
- *SensorMinHwVersion* is the min HW version of the sensors initializated in the system
- *SensorMaxFwVersion* is the max FW version of the sensors initializated in the system
- *SensorMinFwVersion* is the min FW version of the sensors initializated in the system

4.44 Device-Name

- URL: /generalRequest.json
- HTTP methods: PUT/POST
- Required user permission: None
- Required previous command execution: None

This request can be made by all the users.

4.44.1 Request (POST)

```
{
  "VarName": "Device-Name"
}
```

4.44.2 Response (POST)

```
{
  "Value": <String:value>
}
```

The field *Value* is the name of this MACS system.

4.44.3 Request (PUT)

```
{
  "VarName": "Device-Name",
  "Data": {
    "Value": <String:value>
  }
}
```

Where *Value* field is the new name of this MACS system.

4.44.4 Response (PUT)

```
{
  "Result": <I:value>
}
```

5 API v2

In this section there are the descriptions of MACS API v2.

5.1 /v2/simpleLog

- URL: /v2/simpleLog
- HTTP methods: GET, DELETE
- Required user permission: *Read-Log*
- Required previous command execution: None

This request can be made by any user with *Read-Log* permission. It is useful to get the JSON description of system log or to delete it.

5.1.1 GET

The request must be made without any description in the body. The response is like this:

```
{
  {
    "Logs": [
      {
        "Alarm": {
          "Time": <I:value>,
          "OutputId": <I:value>,
          "ZoneId": <I:value>,
          "ZoneName": <String:value>,
          "SensorId": <I:value>,
          "SensorAlarm": <I:bitmask value>

        }
      },
      {
        "Cmd": {
          "Time": <I:value>,
          "Username": <String:value>,
          "Command": <String:value>,
          "Dst": {
            ...
          },
          "Linked": {
            ...
          }
        }
      }
    ]
  }
}
```

Where:

- The log type is indicated through an *Alarm* or a *Cmd* object respectevely for an Alarm or a Command log
- *Time* is the timestamp in which the event happened expressed in unix time
- *OutputId* is the id of the output activated by the event. If it is 0, it is a system alarm

- *ZoneId* is the id of the zone activated by the event. If it is 0, it is a system alarm
- *SensorId* is the id of the sensor activated by the event. If it is 0, it is a system alarm
- *ZoneName* can be:
  - the name of the zone activated
  - the special name of a system alarm
- *SensorAlarm* contains the alarms (one or more) related to the event. See also Alarm value section for a description of alarm values.
- *Username* is the username of the user that trigger the command
- *Command* is the name of the command triggered
- *Dst* is the description of the command or the component modified described as in Data Structure section
- *Linked* is the description of a linked object involved in the command execution

5.2 DELETE

The request must be made without any description in the body. The response is like this:

```
{
  "errorCode": <I:value>
}
```

Where:

- *errorCode* is the error value of the operation (if 0 it is successful)

5.3 /v2/advancedLog

- URL: /v2/adavncedLog
- HTTP methods: GET, DELETE
- Required user permission: *Read-Log*
- Required previous command execution: None

This request can be made by any user with *Read-Log* permission. It is useful to get the JSON description of the advanced log made by sensors or to delete it.

5.3.1 GET

The request must be made without any description in the body. The response is like this:

```
{
  {
    "logs": [
      {
        "Time": <I:value>,
        "OutputId": <I:value>,
        "ZoneId": <I:value>,
        "SensorId": <I:value>,
        "SensorAlarm": <I:bitmask value>
        "ParametersVersion": <I:value>,
        "ConfigurationType": <I:value>,
        "Type": <I: value>,
        "RsrLogs": {
          "Sensitivity": <I:value>,
          "IsResonant": <I:value>,
          "FirstPerturbationCounter": <I:value>,
          "FirstClimbingCounter": <I:value>,
          "FirstClimbingThreshold": <I:value>,
          "StartingPerturbationEnergy": <I:value>,
          "StartingClimbingEnergy": <I:value>,
          "FirstPerturbationResetCounter": <I:value>,
          "FirstClimbingResetCounter": <I:value>,
          "PerturbationEnergies": [
            <I:value>
          ],
          "ClimbingEnergies": [
              <I:value>,
          ]
        }
      },
      {
        "Time": <I:value>,
        "OutputId": <I:value>,
        "ZoneId": <I:value>,
        "SensorId": <I:value>,
        "SensorAlarm": <I:bitmask value>
        "ParametersVersion": <I:value>,
        "ConfigurationType": <I:value>,
        "Type": <I: value>,
        "CutLogs": {
          "Sensitivity": <I:value>,
          "CutDelay": <I:value>,
          "SporadicCutDelay": <I:value>,
          "Events": [
            "Time": <I:value>,
            "Energy": <I:value>,
            "Sensitivity": <I:value>,
            "CutCounter": <I:value>,
            "SporadicCutCounter": <I:value>,
          ]
        }
      }
    ]
  }
}
```

Where:

- *Time* is the timestamp in which the event happened expressed in unix time
- *OutputId* is the id of the output activated by the event. If it is 0, it is a system alarm
- *ZoneId* is the id of the zone activated by the event. If it is 0, it is a system alarm
- *SensorId* is the id of the sensor activated by the event. If it is 0, it is a system alarm
- *zoneName* can be:
  - the name of the zone activated
  - the special name of a system alarm

- *SensorAlarm* contains the alarms (one or more) related to the event. See also Alarm value section for a description of alarm values.
- *ParametersVersion* is the version of parameters
- *ConfigurationType* explain the type of fence
- *Sensitivity* is the value of sensitivity/sensibility of the sensor when event happened
- *IsResonant* is the calibration result of the sensor when event happened
- *FirstPerturbationCounter* is the first value of delay for perturbation algorithm of the sensor
- *FirstClimbingCounter* is the first value of delay for climbing algorithm of the sensor
- *FirstClimbingThreshold* is the first value of threshold for climbing algorithm of the sensor
- *StartingPerturbationEnergy* is the first value of energy for perturbation algorithm of the sensor
- *StartingClimbingEnergy* is the first value of energy for climbing algorithm of the sensor
- *FirstPerturbationResetCounter* is the first value of counter reset for perturbation algorithm of the sensor
- *FirstClimbingResetCounter* is the first value of counter reset for climbing algorithm of the sensor
- *PerturbationEnergies* are the values of perturbation energies frames
- *ClimbingEnergies* are the values of climbing energies frames
- *CutDelay* and *SporadicCutDelay* are the cut delay values of the sensor
- *Events* is an array of the events. The last event is the event which triggers the alarm. Every event is composed as:
  - *Time* is the period of time calculated from the previous event in tens of seconds (the first event time is the time of alarm)
  - *Energy* is the energy calculated by the sensor when the event occurred
  - *Sensitivity* is the sensitivity calculated by the sensor when the event occurred
  - *CutCounter* and *SporadicCutCounter* are the cut counters of the sensor when the event occurred

5.4 DELETE

The request must be made without any description in the body. The response is like this:

```
{
  "errorCode": <I:value>
}
```

Where:

- *errorCode* is the error value of the operation (if 0 it is successful)

6 Deprecated API

In this section there is a list of some old API versions that are not supported anymore

6.1 Log-Prepare

- Deprecated from MACS-ETH v1011 and up
- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Read-Log*
- Required previous command execution: None

This request can be made by any user with *Read-Log* permission. It is useful to require the preparation of a certain type of log.

6.1.1 Request

```
{
  "VarName": "Log-Prepare",
  "Data": {
    "Value": <I:value>
  }
}
```

Where *Value* field can assume the value:

- 1: prepare the alarm log of the system
- 2: prepare the advanced log of sensors

6.1.2 Response

```
{
  "Result": <I:value>
}
```

The response *Result* will be 101 if the request is successful. With this value is requested to the client to continuously request Log-Operation-Status packets to understand if the system finish the log preparation successfully and it is available to receive a new request. If the log preparation was ok, the log file will be available at the link:

- /protected/alarmLog.txt if the alarm log of the system was requested
- /protected/rsrEventsLog.csv if the advanced log of sensors was requested

6.2 Log-Delete

- Deprecated from MACS-ETH v1011 and up
- URL: /generalRequest.json
- HTTP methods: PUT
- Required user permission: *Delete-Log*
- Required previous command execution: None

This request can be made by any user with *Delete-Log* permission. It is useful to delete all log.

6.2.1 Request

```
{
  "VarName": "Log-Delete"
}
```

6.2.2 Response

```
{
  "Result": <I:value>
}
```

The response *Result* will be 101 if the request is successful. With this value is requested to the client to continuously request Log-Operation-Status packets to understand if the system finish the operation successfully and it is available to receive a new request.

6.3 Log-Operation-Status

- Deprecated from MACS-ETH v1011 and up
- URL: /generalRequest.json
- HTTP methods: POST
- Required user permission: *Read-Log*
- Required previous command execution: None

This request can be made by any user with *Read-Log* permission. It can be useful when you wait the result of a log preparation procedure. The client must do this request when the system response to a Log-Prepare or a Log-Delete packet was 101.

6.3.1 Request

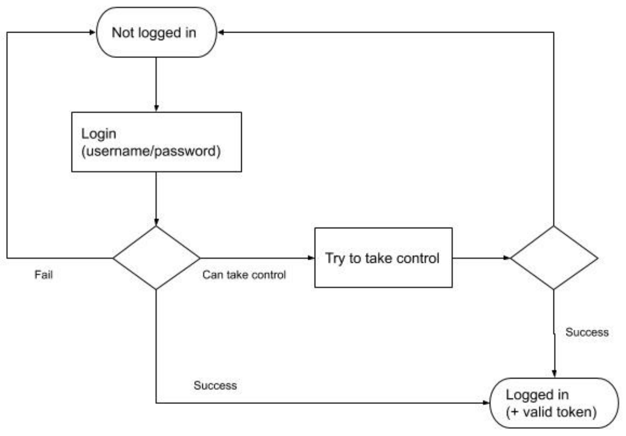
```
{
  "VarName": "Log-Operation-Status"
}
```

6.3.2 Response

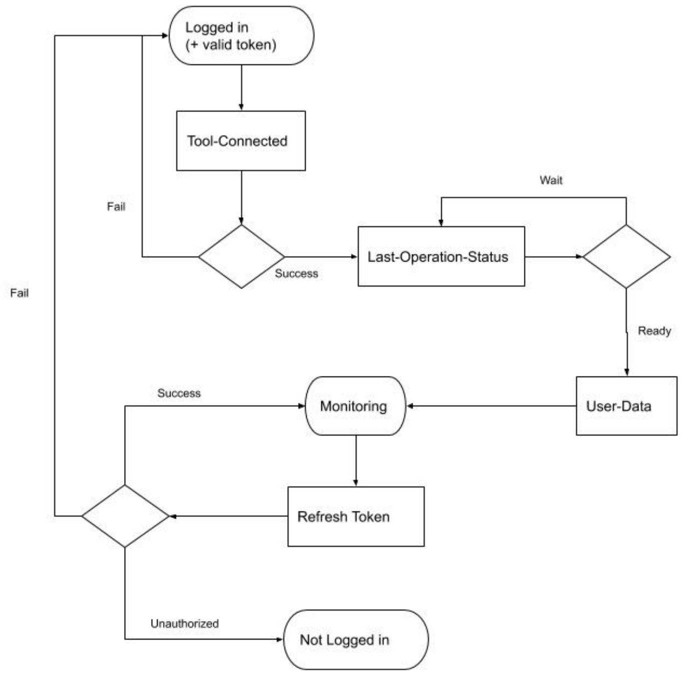
```
{
  "StatusOfOperation": {
    "Status": <I:value>,
    "SystemStatus": <I:value>,
    "TriggeredVarName": <String: value>,
  }
}
```

7 Flowcharts

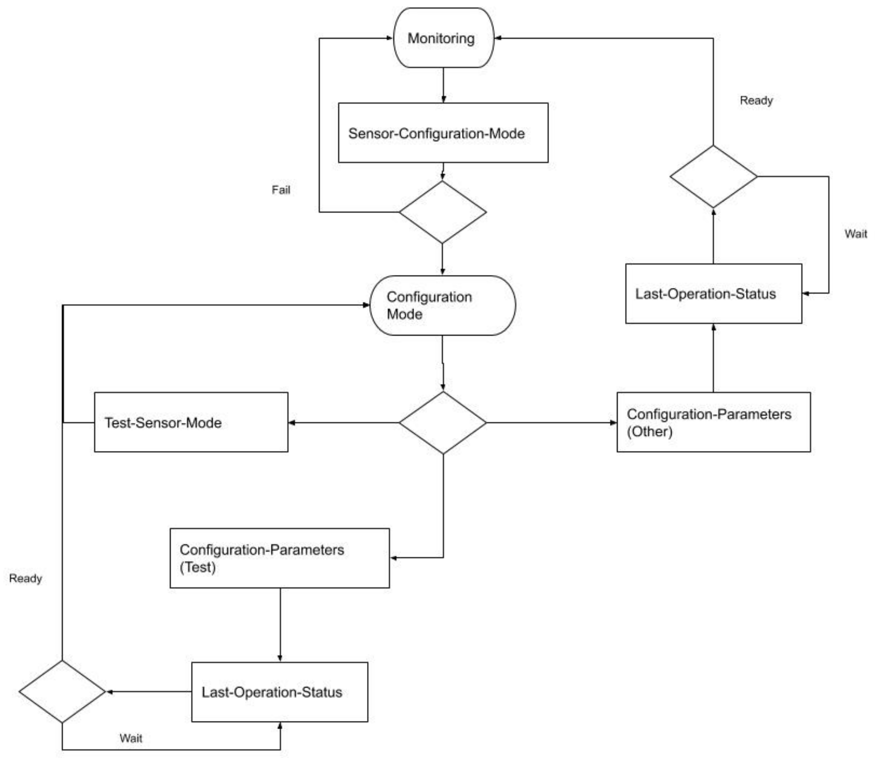
7.1 Login flow



7.2 Monitoring flow



7.3 Configuration flow


Monitoring
Sensor-Configuration-Mode
Fail
Ready
Wait
Last-Operation-Status
Configuration Mode
Test-Sensor-Mode
Configuration-Parameters (Other)
Configuration-Parameters (Test)
Ready
Last-Operation-Status
Wait