IoT Server – REST API

Api Specification Release v.2.1

1 Revisions History

Rev	Date	Description
1.0	2015-07-13	First API v1 version with device configuration API, device configuration variables, real time acquisition of device variables.
1.1	2015-12-21	API v1 extension with system information API, system variables configuration, real time system variables, historic device data and system variables, real time alarms, alarms history.
1.2	2015-02-15	JSON correction example on GET /api/v1/devices/{deviceId}/variables/{variableId}/config.json
1.3	2017-03-29	DRAFT API list/upload/download file
2.0	2018-05-07	Re-organize API published on endpoint v2

2 Introduction

IoT Server provides real time access to its information through API REST of different kind:

- **System Information API** – general system information
- **Devices configuration API** – acquisition of field devices configuration connected to the Alleantia system
- **Variables configuration API** – acquisition of field devices variables configuration, plant / machine data configuration and user defined data
- **Data API** – acquisition of actual and historical field devices variables values, plant / machine data values. Setting of field device variables values (with WRITE permission).
- **Alarms API** – acquisition of actual and historical alarms values configurated in the IoT Server.
- **Events API** – acquisition of actual and historical events values configurated in the IoT Server.

All API returned data representation use the JSON format.

N.B. Suffix *.json* for each endpoint is void. If omitted, returned data format remains JSON.

SYSTEM INFORMATION API HW

Endpoint	HTTP Request
/api/v2/info.json	GET

DEVICES CONFIGURATION API

Endpoint	HTTP Request
/api/v2/devices.json	GET
/api/v2/devices/{deviceId}/config.json	GET

VARIABLES CONFIGURATION API

Endpoint	HTTP Request
/api/v2/devices/{deviceId}/variables/config.json	GET
/api/v2/devices/{deviceId}/variables/{variableId}/config.json	GET
/api/v2/devices/custom/variables/config.json	GET
/api/v2/devices/custom/variables/{variableId}/config.json	GET

DATA API

Endpoint	HTTP Request
/api/v2/devices/{deviceId}/variables/data.json	GET
/api/v2/devices/{deviceId}/variables/{variableId}/data.json	GET
/api/v2/devices/{deviceId}/variables/{variableId}/logdata.json	GET
/api/v2/devices/{deviceId}/variables/{variableId}/set.json	GET
/api/v2/devices/custom/variables/data.json	GET
/api/v2/devices/custom/variables/{variableId}/data.json	GET
/api/v2/devices/custom/variables/{variableId}/logdata.json	GET

PART PROGRAMS API

Endpoint	HTTP Request
/api/v2/devices/{deviceId}/documents/list	GET
/api/v2/devices/{deviceId}/documents/upload	POST
/api/v2/devices/{deviceId}/documents/download	GET
/api/v2/devices/{deviceId}/documents/delete	DELETE
/api/v2/devices/{deviceId}/documents/createSubfolder	POST
/api/v2/devices/{deviceId}/documents/deleteSubfolder	DELETE

ALARMS API

Endpoint	HTTP Request
/api/v2/alarms	GET
/api/v2/alarms/{alarmId}/config.json	GET
/api/v2/alarms/{alarmId}/data.json	GET
/api/v2/alarms/active.json	GET
/api/v2/alarms/history.json	GET

EVENTS API

Endpoint	HTTP Request
/api/v2/events/info.json	GET
/api/v2/events/{eventId}/info.json	GET
/api/v2/events/{eventId}/history.json	GET

HYPERMEDIA LINK

Some responses include hypermedia links used to recover additional information associated to the request. Here follows the hypermedia links structure:

- *rel* – relationship type as defined in RFC 5988, par. 4 “Link relation types”.
- *href* – address to access to recover the information associated to the link.
- *type* (void) – content MIME type.
- *title* (void) – link content description.

3 System information API

3.1 GET /api/v2/info.json

Returns system information through a JSON object with the following fields:

- *sn* – unique serial number
- *uuid* – unique system identifier (alphanumeric)
- *producer* – producer
- *name* – system name (user defined)
- *localDate* – actual system date
- *webAppVersion* – system software version

3.1.1 URL Parameters

none

3.1.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/info.json
Body	*void*

3.1.3 Sample Reply

Status code	200 OK

Body	{
	"producer": "Alleantia",
	"sn": "IOTSPI215060600",
	"uuid": "414da140b-1ead-32f8-ae78-adf800fcdc96",
	"name": "Monitoraggio impianto",
	"localDate": "2015-11-12T17:56:34.831+0100",
	"webAppVersion": "4.1.7"
	}

4 Devices configuration API

4.1 GET /api/v2/devices.json

Returns a data list to retrieve information on all devices configured in the IoT Server. The list contains zero or more JSON objects with the following fields:

- *id* – device numerical identifier. It is a unique identifier only within a specific ISS system and is only valid for the system that has returned it. If the user subsequently modifies the IoT Server configuration, by disabling or removing the device, further calls to this API will no longer provide the device id.
- *links* – list of hypermedia links to recover from the system the connected devices and variables configuration information; links already include the deviceId parameters required for subsequent calls. The "rel" parameter identifies the information type returned by the link:
  - *config-device* - device configuration
  - *config-variables* - device variables configuration

The returned list is void if no devices are configured in the system.

4.1.1 URL Parameters

none

4.1.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices.json
Body	*void*

4.1.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return
Body	[{ "id": 1, "links": [{ "rel": "config-device", "href": "http://192.168.1.29/api/v2/devices/1/config", "title": "Device configuration" }, { "rel": "config-variables", "href": "http://192.168.1.29/api/v2/devices/1/variables/config",

```
            "title": "Device variables configuration"
          }
        ]
      },
      {
        "id": 7,
        "links": [
          {
            "rel": "config-device",
            "href": "http://192.168.1.29/api/v2/devices/7/config",
            "title": "Device configuration"
          },
          {
            "rel": "config-variables",
            "href": "http://192.168.1.29/api/v2/devices/7/variables/config",
            "title": "Device variables configuration"
          }
        ]
      }
    ]
```

4.2 GET /api/v2/devices/{deviceId}/config.json

Returns the configuration information for device configured in the system; the returned data is a JSON object containing the following fields:

- *id* – device numerical identifier. Same rules applies as for the previous request.
- *manufacturer* – name of device manufacturer.
- *model* – device model.
- *version* (optional) – device version.
- *description* – device description (IoT Server user defined).
- *category* – device category.
- *links* (optional) – list of hypermedia links to recover documents associated to the device.

If the device does not exist, API returns status code "204 - No content" with void body.

4.2.1 URL Parameters

deviceId	Device numerical identifier

4.2.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/1/config.json
Body	*void*

4.2.3 Sample Reply

Status code	200 Ok 404 Not Found – Device does not exist

Body

```
{
  "id": 1,
  "manufacturer": "Fronius",
  "model": "IG Plus 120 V-3",
  "description": "INVERTER 1",
  "category": "Inverter",
  "links": [
    {
      "rel": "edit-media",
      "href": "http://192.168.1.29/documents/34.pdf",
      "type": "application/pdf",
      "title": "Service manual"
    }
  ]
}
```

5 Variables configuration API

Used to obtain the device variables configuration (by specifying a deviceId) or for a full plant (multiple devices). This API returns a JSON object (or a list of JSON objects) containing the following fields:

- *id* – numerical identifier for the device or plant variable. Device variable Id is unique only within the specific deviceId and is only valid for the IoT Server system that has returned it. Plant variable Id is unique only within the specific IoT Server that has returned it. If the user subsequently modifies the IoT Server configuration, by disabling the device variable, further calls to this API will no longer provide the id
- *description* – variable description, assigned by the device producer or the IoT Server user customization.
- *datatype* – variable type, values can be “String”, “Boolean” or “Numeric”.
- *engUnit* (optional) – variable unit of measure.
- *minimum* (optional) – min value that the variable can assume; it is assigned by the manufacturer or IoT Server user.
- *maximum* (optional) – max value that the variable can assume; it is assigned by the manufacturer or IoT Server user.
- *link* – hypermedia link to recover the actual variable value. “rel” parameter value is “data”.
- *alarmable* – boolean that indicates the existence of an alarm associated to the variable.
- *writable* – boolean that indicates if the variable is writable or not.

5.1 GET /api/v2/devices/{deviceId}/variables/config.json

Returns a list with the information on variables configuration of a device configured in the IoT Server.

5.1.1 URL Parameters

deviceId	Device numerical identifier

5.1.2 Query Parameters

id (optional)	Variable Id used to filter results

5.1.3 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/1/variables/config.json
Sample URL	http://192.168.1.29/api/v2/devices/1/variables/config.json?id=1&id=2
Body	*void*

5.1.4 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return 404 Not Found – Device does not exist

Body


```
[  {
     "id": 1,
     "description": "Power - NOW",
     "dataType": "Numeric",
     "engUnit": "kW",
     "minimum": 0,
     "maximum": 10,
     "link": {
       "rel": "data",
       "href": "http://192.168.1.29/api/v2/devices/1/variables/1/data"
     },
     "alarmable": false,
     "writable": false
   },
   {
     "id": 2,
     "description": "Energy - TOTAL",
     "dataType": "Numeric",
     "engUnit": "kWh",
     "link": {
       "rel": "data",
       "href": "http://192.168.1.29/api/v2/devices/1/variables/2/data"
```

```
        },
        "alarmable": false,
        "writable": false
      }  ]
```

5.2 GET /api/v2/devices/{deviceId}/variables/{variableId}/config.json

Returns the information on a variable configuration for a device configured on the IoT Server.

If the device or the variable do not exist, API returns status code “204 - No content” with void body.

5.2.1 URL Parameters

deviceId	Device numerical identifier
variableId	Numeric Id for the device variable

5.2.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/1/variables/9/config.json
Body	*void*

5.2.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return 404 Not Found – Device does not exist

Body

```
{
  "id": 9,
  "description": "DC voltage - NOW",
  "dataType": "Numeric",
  "engUnit": "V",
  "minimum": 230,
  "maximum": 600,
  "link": {
      "rel": "data",
      "href": "http://192.168.1.29/api/v2/devices/1/variables/9/data"
    },
    "alarmable": false,
    "writable": false
}
```

5.3 GET /api/v2/devices/custom/variables/config.json

Returns a list with the information on variables configuration of the plant configured in the IoT Server.

5.3.1 URL Parameters

none

5.3.2 Query Parameters

id (optional)	Variable Id used to filter results

5.3.3 Sample Reply

Sample URL	http://192.168.1.29/api/v2/devices/custom/variables/config.json
Sample URL	http://192.168.1.29/api/v2/devices/custom/variables/config.json?id=5&id=500
Body	*void*

5.3.4 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return

Body


```
[
  {
    "id": 5,
    "description": "Potenza istantanea",
    "dataType": "Numeric",
    "engUnit": "kW",
    "link": {
      "rel": "data",
      "href":
"http://192.168.1.29/api/v2/devices/custom/variables/5/data"
    },
    "alarmable": false,
    "writable": false
  },
  {
    "id": 500,
    "description": "Integrale Energia",
    "dataType": "Numeric",
    "engUnit": "kWh",
    "link": {
      "rel": "data",
      "href":
"http://192.168.1.29/api/v2/devices/custom/variables/50000/data"
    },
    "alarmable": false,
    "writable": false
```

```
        }
    ]
```

5.4 GET /api/v2/devices/custom/variables/{variableId}/config.json

Returns the information on a variable configuration for a plant configured on the IoT Server.

If the variable do not exist, API returns status code “204 - No content” with void body.

5.4.1 URL Parameters

variableId	Numeric id for the plant variable

5.4.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/custom/variables/500/config.json
Body	*void*

5.4.3 Sample Reply

Status code	200 Ok 404 Not Found – Variable does not exist

Body	{
	"id": 500,
	"description": "Integrale Energia",
	"dataType": "Numeric",
	"engUnit": "kWh",
	"link": {
	"rel": "data",
	"href":
	"http://192.168.1.29/api/v2/devices/custom/variables/50000/data"
	},
	"alarmable": false,
	"writable": false
	}

6 Data API

For all these API the returned type is a JSON object containing the following fields:

- *id* – numeric identifier for the device or plant variable.
- *value* - variable value; types can be can be “String”, “Boolean” or “Numeric”.
- *decodedValue* (optional) – where variable values codes for which a decoding is provided to improve the "readability" to the end user (i.e. 0 = OK, 1 = FAULT), this field contains the value field decoded.
- *timestamp* – date and time in ISO-8601 format when value was recorded.
- *quality* - boolean value indicating whether the last reading attempt of the variable was successful. It indicates if the value field contains a current or potentially obsolete information. If the last.

If the response has no errors, body contains JSON object with status code è 200 OK; if the variable requested do not exist in the IoT System configuration, 2 responses are possible depending on endpoint return type:

- *Endpoint with return type single data* – returns a status code “204 - No content” with void body.
- *Endpoint with return type list* – returns a status code 200 OK and a void list.

N.B. In case the requested variable is not yet available (when a communication between the IoT Server
and the device has not yet occurred), the timestamp field has null value and the value field has a default value that the requesting application MUST ignore since the data was not actually collected from the device.

6.1 GET /api/v2/devices/{deviceId}/variables/data.json

Returns a list of actual values for the variables of a device configured in the IoT Server.

6.1.1 URL Parameters

deviceId	Device numerical identifier

6.1.2 Query Parameters

id (optional)	Variable Id used to filter results

6.1.3 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/5/variables/data.json
Sample URL	http://192.168.1.29/api/v2/devices/5/variables/data.json?id=24&id=27
Body	*void*

6.1.4 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return 404 Not Found – Device does not exist

Body

```
[
  {
    "id": 24,
    "value": true,
    "timestamp": "2015-02-13T16:17:42.831+0100",
    "quality": true
  },
  {
    "id": 27,
    "value": 12.76,
    "timestamp": "2015-02-13T16:17:41.299+0100",
    "quality": true
  }
]
```

6.2 GET /api/v2/devices/{deviceId}/variables/{variableId}/data.json

Returns the actual variable value for a device configured in the IoT Server.

6.2.1 URL Parameters

deviceId	Device numerical identifier
variableId	Numeric Id for the device variable

6.2.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/5/variables/3/data.json
Body	*void*

6.2.3 Sample Reply

Status code	200 OK

Body

```
{
  "id": 3,
  "value": 2561,
  "timestamp": "2015-02-11T15:31:02.404+0100",
  "decodedValue": "FAN FAULT",
  "quality": false
}
```

6.3 GET /api/v2/devices/{deviceId}/variables/{variableId}/set.json

Set the value of a writable variable for a device configured in the IoT Server.

Response occur if request has been accepted.

NB: request accepted DO NOT imply request completed! Best practice suggests to execute a read request to verify if and when the command was executed.

6.3.1 URL Parameters

deviceId	Device numerical identifier
variableId	Numeric Id for the device variable
value	New value for the variable

6.3.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/11/variables/7/set?value=-54.34
Body	*void*

6.3.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return 400 Bad Request – Mandatory data is missing 401 Unauthorized – License not enabled to the operation 403 Forbidden – Not enabled/writable variable 404 Not Found – Device does not exist

Body	{
	"accepted": true,
	"description": "Accepted"
	}

6.4 GET /api/v2/devices/{deviceId}/variables/{variableId}/logdata.json

Return the historical values of a variable of a device configured in the IoT Server for a specified time interval. Returns a void body if there is no data in the specified time interval. The maximum number of returned values is limited to the first 1000.

6.4.1 URL Parameters

deviceId	Device numerical identifier
variableId	Numeric Id for the device variable
startTime	Start time for the time interval requested (integer, milliseconds from 1.1.1970 00:00:00.000)
endTime (optional)	End time for the time interval requested (integer, milliseconds from 1.1.1970 00:00:00.000). If not specified, upper limit is current time.

6.4.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/3/variables/3/logdata?startTime=0
Body	*void*

6.4.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return 400 Bad Request – Mandatory data is missing 404 Not Found – Device does not exist

Body

```
[
  {
    "value": 1,
    "timestamp": "2015-11-12T16:14:01.713+0100",
    "quality": true
  },
  {
    "value": 3,
    "timestamp": "2015-11-12T16:15:07.162+0100",
    "quality": true
  },
…
```

6.5 GET /api/v2/devices/custom/variables/data.json

Returns a list with the variables actual values for the plant configured in the IoT Server.

6.5.1 URL Parameters

none

6.5.2 Query Parameters

id (optional)	Variable Id used to filter results

6.5.3 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/custom/variables/data.json
Sample URL	http://192.168.1.29/api/v2/devices/custom/variables/data.json?id=5&id=500
Body	*void*

6.5.4 Sample Reply

Status code	200 Ok 204 No Content –Nothing to return

Body

```
[
  {
    "id": 5,
    "value": 0,
    "timestamp": "2015-11-16T10:29:35.182+0100",
    "quality": false
  },
  {
    "id": 500,
    "value": 2.2757686111111113,
    "timestamp": "2015-11-16T10:29:55.959+0100",
    "quality": false
  }
]
```

6.6 GET /api/v2/devices/custom/variables/{variableId}/data.json

Returns the actual variable value for the plant configured in the IoT Server.

6.6.1 URL Parameters

variableId	Numeric identification of the system variable

6.6.2 Sample Reply

Sample URL	http://192.168.1.29/api/v2/devices/custom/variables/50000/data.json
Body	*void*

6.6.3 Sample Reply

Status code	200 Ok 404 Not Found – Variable does not exist

Body	{
	"id": 50000,
	"value": 2.2757686111111113,
	"timestamp": "2015-11-16T10:31:47.288+0100",
	"quality": false
	}

6.7 GET /api/v2/devices/custom/variables/{variableId}/logdata.json

Return the historical values of a plant variable configured in the system for a specified time interval. Returns a void body if there is no data in the specified time interval. The maximum number of returned values is limited to the first 1000.

6.7.1 URL Parameters

variableId	Numeric identification of the system variable
startTime	Start time for the time interval requested (integer, milliseconds from 1.1.1970 00:00:00.000)
endTime (optional)	End time for the time interval requested (integer, milliseconds from 1.1.1970 00:00:00.000). If not specified, upper limit is current time.

6.7.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/custom/variables/500/logdata?startTime=0
Body	*void*

6.7.3 Sample Reply

Status code	200 Ok 400 Bad Request – Mandatory data is missing 404 Not Found –Variable does not exist

Body

```
[
  {
    "value": 0.05625666666666667,
    "timestamp": "2015-11-12T16:14:02.272+0100",
    "quality": true
  },
  {
    "value": 0.10433305555555555,
    "timestamp": "2015-11-12T16:15:07.367+0100",
    "quality": true
  },
…
```

7 Management of device-connected files and folders

7.1 GET /api/v2/devices/documents/list

Returns the list of files and directories (aliases and real paths) for every device connected to the IoT. It's possible to execute operations on them through API documents.

As a convention, paths use as a separator character "/". In the case of a path that references a directory, this needs to end with the "/" character.

7.1.1 URL Parameters

none

7.1.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/documents/list
Body	*void*

7.1.3 Sample Reply

Status code	200 Ok 204 No content – If there are no directories

Body

```
[
    {
        "devId": 179,
        "description": "Fanuc32i",
        "list": [
            {
                "name": "PATH1",
                "path": "//CNC_MEM/USER/PATH1/",
                "type": "FOLDER",
                "read": true,
                "write": true,
                "create": true,
                "delete": true
            }
        ]
    },
    {
        "devId": 180,
        "description": "Heidenhain640",
        "list": [
            {
                "name": "ROOT",
                "path": "TNC:\\",
                "type": "FOLDER",
                "read": true,
                "write": true,
                "create": true,
                "delete": true
            }
```

```
                ]
            }
        ]
```

7.2 GET /api/v2/devices/{deviceId}/documents/list

Returns the list of files and directories (aliases and real paths) in the indicated folder, if it's authorized.

As a convention, paths use as a separator character "/". In the case of a path that references a directory, this needs to end with the "/" character.

N.B: The "path" parameter is optional. If it's missing, the API will return the list of every readable folder.

7.2.1 URL Parameters

deviceId	Numeric id for the device
path	Path of the directory of the content.

7.2.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/1/documents/list?path=ROOT/nc_prog/
Body	*void*

7.2.3 Sample Reply

Status code	200 Ok 204 No content – If there are no directories 400 Bad Request – Missing or wrong path 403 Forbidden – No privileges for the operation 404 Not Found – Device not found 406 Not acceptable – Device not enabled to part programs exchange 500 Internal Server Error – Generic error

Body


```
[
    {
        "name": "nc_prog",
        "type": "folder",
        "size": 0,
        "lastModified": "2018-03-29T12:19:43.000+0200",
        "links": [
            {
                "title": "Folder List",
                "rel": "documents",
                "href":
"http://localhost:8080/api/v2/devices/162/documents/list?path=ROOT/nc_prog
/"
            },
```

```
                {
                    "title": "File Upload",
                    "rel": "documents",
                    "href":
"http://localhost:8080/api/v2/devices/162/documents/upload?destinationFold
er=ROOT/nc_prog/{FILENAME}"
                },
                {
                    "title": "Folder Create",
                    "rel": "documents",
                    "href":
"http://localhost:8080/api/v2/devices/162/documents/createSubfolder?folder
Path=ROOT/nc_prog/{FOLDERNAME}"
                },
                {
                    "title": "Folder Delete",
                    "rel": "documents",
                    "href":
"http://localhost:8080/api/v2/devices/162/documents/deleteSubfolder?folder
Path=ROOT/nc_prog/"
                }
            ]
        },…
]
```

7.3 POST /api/v2/devices/{deviceId}/documents/upload

Upload a file in the selected folder.

The Content-Type used by the service is multipart/form-data.

As a convention, paths use as a separator character "/". In the case of a path that references a directory, this needs to end with the "/" character.

Set in the request's body an entry with key=uploadFile and the file to send as a value.

7.3.1 URL Parameters

deviceId	Numeric id for the device
destinationFolder	Folder path in which to upload the file

7.3.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/11/documents/upload? destinationFolder=ROOT/nc_prog/
Body	*Key=uploadFile – Value(File)=File to upload*

7.3.3 Sample Reply

Status code	200 Ok 400 Bad Request – Missing or wrong path 403 Forbidden – No privileges for the operation 404 Not Found – Device not found 406 Not acceptable – Device not enabled to part programs exchange 500 Internal Server Error – Generic error

Body	{
	"accepted": true,
	"description": "File uploaded successfully"
	}

7.4 GET /api/v2/devices/{deviceId}/documents/download

Download the selected file.

As a convention, paths use as a separator character "/". In the case of a path that references a directory, this needs to end with the "/" character.

7.4.1 URL Parameters

deviceId	Numeric identification of the device
filePath	Folder path of the file to download

7.4.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/11/documents/download?filePath=ROOT/nc_prog/file.h
Body	*void*

7.4.3 Sample Reply

Status code	200 Ok 400 Bad Request – Missing or wrong path 403 Forbidden – No privileges for the operation 404 Not Found – Device not found 406 Not acceptable – Device not enabled to part programs exchange 500 Internal Server Error – Generic error

Body	{
	"accepted": true,
	"description": "File downloaded successfully"
	}

7.5 DELETE /api/v2/devices/{deviceId}/documents/delete

Deletes the selected file.

As a convention, paths use as a separator character "/". In the case of a path that references a directory, this needs to end with the "/" character.

7.5.1 URL Parameters

deviceId	Numeric identification of the device
filePath	Path of the file to delete

7.5.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/11/documents/delete?filePath=ROOT/nc_prog/file.h
Body	*void*

7.5.3 Sample Reply

Status code	200 Ok 400 Bad Request – Missing or wrong path 403 Forbidden – No privileges for the operation 404 Not Found – Device not found 406 Not acceptable – Device not enabled to part programs exchange 500 Internal Server Error – Generic error

Body	{
	"accepted": true,
	"description": "File deleted successfully"
	}

7.6 POST /api/v2/devices/{deviceId}/documents/createSubfolder

Create a sub-folder in the selected path.

As a convention, paths use as a separator character "/". In the case of a path that references a directory, this needs to end with the "/" character.

7.6.1 URL Parameters

deviceId	Numeric identification of the device
folderPath	Path of the folder in which to create the new folder
subfolderName	New folder name

7.6.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/11/documents/createSubfolder?folderPath=ROOT/nc_prog/&subfolderName={FOLDERNAME}
Body	*void*

7.6.3 Sample Reply

Status code	200 Ok 400 Bad Request – Missing or wrong path 403 Forbidden – No privileges for the operation 404 Not Found – Device not found 406 Not acceptable – Device not enabled to part programs exchange 500 Internal Server Error – Generic error

Body	{
	"accepted": true,
	"description": "Folder created successfully"
	}

7.7 DELETE /api/v2/devices/{deviceId}/documents/deleteSubfolder

Delete a sub-folder in the selected path.

As a convention, paths use as a separator character "/". In the case of a path that references a directory, this needs to end with the "/" character.

7.7.1 URL Parameters

deviceId	Numeric identification of the device
folderPath	Path of the folder to delete

7.7.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/devices/11/documents/createSubfolder?folderPath = ROOT/nc_prog/
Body	*void*

7.7.3 Sample Reply

Status code	200 Ok 400 Bad Request – Missing or wrong path 403 Forbidden – No privileges for the operation 404 Not Found – Device not found 406 Not acceptable – Device not enabled to part programs exchange 500 Internal Server Error – Generic error

Body	{
	"accepted": true,
	"description": "Folder deleted successfully"
	}

8 Alarms API

8.1 GET /api/v2/alarms.json

Returns a list of all active alarms in the IoT Server. In case there are no active alarms returns a void list

The returned type is a list of JSON objects containing the following fields:

- *id* – alarm numeric id.
- *links* – hypermedia link to recover the configuration of a single alarm. "rel" parameter has "config-alarm" value.

8.1.1 Sample Request

Sample URL	http://192.168.1.29/api/v2/alarms.json
Body	*void*

8.1.2 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return

Body	[
	{
	"id": 5,
	"links": [
	{
	"rel": "config-alarm",
	"href": "http://192.168.1.29/api/v2/alarms/5/config",
	"title": "Alarm configuration"
	}
	]
	},
	{
	"id": 1,
	"links": [
	{
	"rel": "config-alarm",
	"href": "http://192.168.1.29/api/v2/alarms/1/config",
	"title": "Alarm configuration"
	}
	]
	}
	]

8.2 GET /api/v2/alarms/{alarmId}/config.json

Returns the information on an alarm configured on the IoT Server.

If the requested alarm does not exist, API returns status code “204 - No content” with void body.

The returned type is a JSON object with the following fields:

- *id* – alarm numeric Id.
- *name* – alarm name.
- *description* – alarm description.
- *link* – hypermedia link to recover the actual alarm value. “rel” parameter has value “data”.
- *snapshotGlobalIds(optional)* – list of globalId referenced by the alarm (no custom alarms).

8.2.1 URL Parameters

alarmId	Alarm numeric Id

8.2.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/alarms/4/config.json
Body	*void*

8.2.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return 404 Not found – Can’t compute Alarm

Body

```
{
  "id": 4,
  "name": "Allarme Sensore 3",
  "description": "Input channel 3 sensor error",
  "link": {
    "rel": "data",
    "href":
      "http://192.168.1.29/api/v2/alarms/4/data",
    "title": "Alarm value"
  }
}
```

8.3 GET /api/v2/alarms/active.json

Returns a list of active alarms in the IoT Server.

If the system does not have active alarms the returned list is empty. The returned type is a list of JSON objects with the following fields:

- *id* – alarm numeric id.
- *eventId* (optional)– event identifier (integer; identifies a pair for activation and de-activation time of an alarm event).
- *deviceName* (optional) – device associated to the alarm.
- *deviceSection* (optional) – device section associated to the alarm.
- *measure* (optional) – variable to which the alarm is associated.
- *description* – alarm description.
- *onDate* – alarm activation timestamp.
- *quality* – boolean value that indicates if the last reading attempt was successful
- *alarmed* – alarm state

8.3.1 Query Parameters

alarmId (optional)	Variable id to filter results

8.3.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/alarms/active?alarmId=5&alarmId=7&alarmId=9
Body	*void*

8.3.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return

Body


```
[
  {
    "id": 5,
    "eventId": 166050,
    "deviceName": "Curr",
    "measure": "Curr 1",
    "description": "Test Curr",
    "onDate": "2015-11-16T11:01:42.883+0100",
    "quality": false,
    "alarmed": false
  },
  {
    "id": 7,
```

```
      "eventId": 166051,
      "measure": "Potenza istantanea",
      "description": "Test VV",
      "onDate": "2015-11-16T11:01:57.462+0100",
      "quality": false,
      "alarmed": false
    },
    {
      "id": 9,
      "eventId": 166049,
      "description": "Test multi",
      "onDate": "2015-11-16T11:01:42.769+0100",
      "quality": false,
      "alarmed": false
    },
  ]
```

8.4 GET /api/v2/alarms/history.json

Returns the historical list of the alarms in the IoT Server sorted by ascending time. If there are no alarms, returns an empty list. Each item in the list is identified by an eventId; it includes an activation time and (if any) a deactivation time. It is possible to specify an eventId range to select only the included alarms. The number of returned alarms is limited to the first 1000 results.
The type returned is a list of JSON objects with the following fields:

- *alarmType* – Alarm type; can be "DEVICE_ALARM" for alarms associated to a device; "SYS_ALARM" for the system alarms; "ALARM" for other *alarms.*
- *alarmId (optional)* – Alarm identifier (only if alarmtype is "ALARM").
- *eventId* – Event identifier (integer, identifies a pair for activation and de-activation time for an alarm).
- *onDate* – alarm activation timestamp.
- *offDate* (optional) – alarm deactivation timestamp.
- *deviceName* (optional) – device associated to the alarm.
- *deviceSection* (optional) – device section associated to the alarm.
- *description –alarm description.*

8.4.1 URL Parameters

startEventId (optional)	Identifier for the first event (included) to start retrieving alarms. If unspecified, starts with the first alarm.
endEventId (optional)	Identifier for the last event (included) to retrieve. If unspecified, ends with the last alarm.
includeActive (optional)	Default: "false": it returns only historical alarms. If "true" returns both active and historical alarms.

8.4.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/alarms/history.json
Sample URL	http://192.168.1.29/api/v2/alarms/history?startEventId=166134&endEventId=166226&includeActive=true
Body	*void*

8.4.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return

Body

```
[
  {
    "alarmType": "ALARM",
    "alarmId": 9,
    "eventId": 166134,
    "onDate": "2015-11-27T17:10:29.451+0100",
    "deviceName": "Tool",
    "deviceSection": "B1",
    "description": "Tool 3 Alarm"
  },
  {
    "alarmType": "DEVICE_ALARM",
    "eventId": 166135,
    "onDate": "2015-11-30T09:15:03.210+0100",
    "deviceName": "Tool",
    "description": "1"
  },
  {
    "alarmType": "DEVICE_ALARM",
    "eventId": 166226,
    "onDate": "2015-12-17T10:59:15.642+0100",
    "offDate": "2015-12-17T11:30:29.857+0100",
    "deviceName": "MB Write Test",
    "description": "1"
  },
]
```

9 Events API

9.1 GET /api/v2/events/info.json

Return information relative to events that can be generated by the system through a JSON object with the following fields:

- *eventId* – event unique id
- *eventName* – event name
- *condition* – event triggering condition
- *delayOn* – delay to apply before marking the event as active
- *delayOff* – delay to apply before marking the event as deactivated
- *type* – type of event
- *snapshotGlobalIds* – id list of the monitored variable at the moment of the event triggering
- *comparisonOperator* – (optional) operator for the creation of a boolean condition.
- *numericCompareValue* – (optional) value to make a boolean confrontation.

9.1.1 URL Parameters

none

9.1.2 Sample Request

Sample URL	http://192.168.1.29/ api/v2/events/info.json
Body	*void*

9.1.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return
Body	[{ "eventId": 2, "eventName": " Evento2", "condition": "$G_21_47", "delayOn": 0, "delayOff": 0, "type": "onChange", "snapshotGlobalIds": "G_21_803,G_21_20", "comparisonOperator": "eq", "numericCompareValue": 0

```
    },
    {
        "eventId": 1,
        "eventName": "Evento1",
        "condition": "($G_21_47) eq true",
        "delayOn": 0,
        "delayOff": 0,
        "type": "boolean",
        "snapshotGlobalIds": ["G_21_803","G_21_20"],
        "comparisonOperator": "eq",
        "numericCompareValue": 0
    }
]
```

9.2 GET /api/v2/events/{eventId}/info.json

Return information relative to the single event that can be generated by the system through a JSON object with the following fields:

- *eventId* – event unique id
- *eventName* – event name
- *condition* – event triggering condition
- *delayOn* – delay to apply before marking the event as active
- *delayOff* – delay to apply before marking the event as deactivated
- *type* – type of event
- *snapshotGlobalIds* – id list of the monitored variable at the moment of the event triggering
- *comparisonOperator* – operator for the creation of a boolean condition
- *numericCompareValue* – value to make a boolean confrontation

9.2.1 URL Parameters

eventId	event numeric id

9.2.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/info.json
Body	*void*

9.2.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return 404 Not Found – Event not found

Body


```
{
    "eventId": 1,
    "eventName": " Evento1",
    "condition": "$G_21_47",
    "delayOn": 0,
    "delayOff": 0,
    "type": "onChange",
    "snapshotGlobalIds": ["G_21_803","G_21_20"],
    "comparisonOperator": "eq",
    "numericCompareValue": 0
}
```

9.3 GET /api/v2/events/{eventId}/history.json

Return the historical events in the IoT Server for a specified time interval. Returns a void body if there is no data in the specified time interval. The maximum number of returned values is limited to the first 1000.

9.3.1 URL Parameters

eventId	Event numeric id
startTime	Start time for the time interval requested (integer, milliseconds from 1.1.1970 00:00:00.000)
endTime (optional)	End time for the time interval requested (integer, milliseconds from 1.1.1970 00:00:00.000). If not specified, upper limit is current time.

9.3.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/events/1/history?startTime=0
Body	*void*

9.3.3 Sample Reply

Status code	200 Ok 204 No Content – Nothing to return 400 Bad Request – Mandatory data is missing 404 Not Found – Event not found

Body


```
[
  {
    "eventId": 3,
    "eventName": "Evento1",
    "timestamp": "2017-07-31T15:37:53.941+0200",
    "state": true,
    "variablesSnapshot": [
      {
        "devId": 21,
        "varId": 803,
        "quality": true,
        "value": true
      },
      {
        "devId": 21,
        "varId": 20,
        "quality": true,
        "value": 514000
```

```
            }
         ]
      },
      {
         "eventId": 3,
         "eventName": "Evento1",
         "timestamp": "2017-07-31T15:38:36.301+0200",
         "state": false,
         "variablesSnapshot": [
            {
               "devId": 21,
               "varId": 803,
               "quality": true,
               "value": true
            },
            {
               "devId": 21,
               "varId": 20,
               "quality": true,
               "value": 514000
            }
         ]
      },
      …
   ]
```

10 License

10.1 GET /api/v2/license/info.json

Return the device serial, useful for license activation and to know its state. If the software is licensed, it will return a report of the enabled functionalities.

10.1.1 URL Parameters

none

10.1.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/license/info
Body	*void*

10.1.3 Sample Reply

Status code	200 Ok

Body


```
{
  "devSn": "49ab91753-9b94-3d2e-8ff6-17d3d5113606",
  "licensed": true,
  "deviceCount": 15,
  "variableCount": 5000,
  "synopticCount": 6,
  "logDaysCount": 90,
  "iothubServicesCount": 2,
  "sqlServicesCount": 2,
  "dropboxServicesCount": 2,
  "onedriveServicesCount": 2,
  "modbusGWEnabled": true,
  "modbusWriteEnabled": true,
  "restAPIEnabled": true,
  "restAPIWriteEnabled": true,
  "energyPackEnabled": true,
  "machinePackEnabled": false,
  "navyPackEnabled": true,
  "yammerEnabled": true
}
```

10.2 POST /api/v2/license/activate.json

Activate, if possible, the license obtained using the file license.lic as a parameter in the message body. It's possible to activate the license only once; currently it's not possible to upgrade it through an API Rest call.

Set in the request's body an entry with key=uploadFile and the file to send as a value.

10.2.1 URL Parameters

reboot (optional)	Boolean true/false to determinate if to reboot the software after the license's activation (needed to enable the full licensed functionality).

10.2.2 Sample Request

Sample URL	http://192.168.1.29/api/v2/license/activate?reboot=true
Body	*Key=license – Value(File)=File to upload*

10.2.3 Sample Reply

Status code	200 Ok 403 Forbidden – If a re-activation/upgrade on the license is tried 500 Internal Error – If there is a generic error during the activation

Body	{
	"accepted": true,
	"description": "License loaded. Rebooting..."
	}