IoT-TICKET®

YOUR TICKET TO INTERNET OF THINGS AND BEYOND

IoT-TICKET USER MANUAL

REST API

Document version: IoT-TICKET 3.19.1

Contents

Table of tables

Table of figures

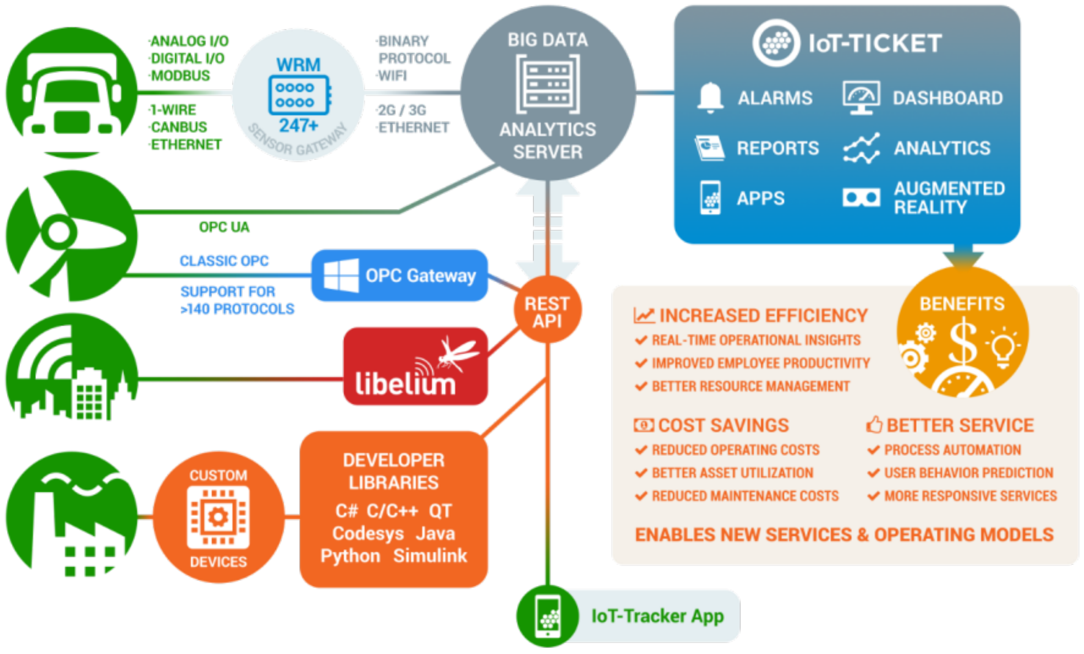
1 IoT-TICKET REST API

About IoT-TICKET®

IoT-TICKET is your Ticket to the Internet of Things and beyond… Developed since 2005, IoT-TICKET is, with over 1.6 million users, one of the most advanced and complete Internet of Things platforms on the market.

1 Most advanced Internet of Things platform

IoT-TICKET covers versatile data-acquisition needs, Big-Data and analytics enabled servers, web-based Dashboards and Reports. With IoT-TICKET, one can create and deploy fully fledged IoT applications in minutes.



2 IoT-TICKET benefits

2 Introduction

This document provides the guidelines for the WRM Internet of Things Rest API. It explains how devices are created and managed, how data nodes are created, written to and read from. While the examples provided are not exhaustive in scope, they do provide a clear indication of what is expected as input and output to the API.

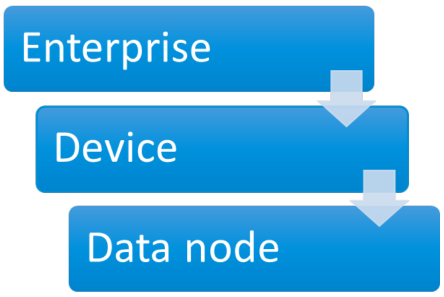
2.1 Abbreviations and definitions

Definition	Explanation
REST	Representational state transfer
API	Application Programming Interface
JSON	JavaScript Object Notation
XML	Extensible Markup Language
IoT	Internet of Things

In the document, an integer type refers to a 32 bit signed integer (2^{31}-1 to -2^{31}) and a long is a 64 bit signed integer (2^{63}-1 to -2^{63}). The length of strings accepted as parameter to a call are stated in the form ***String:(max-length)*** meaning the string should not exceed ***max-length*** characters. Only UTF-8 encoded strings are accepted.

2.2 Data Model

The WRM system data model visible through this REST API is illustrated in the figure below.



3 System data model

At the base of this model, *Enterprises* are situated (e.g. companies or customers). Each Enterprise can have multiple devices that need to be monitored (e.g. a home, an engine, or a truck). A device can have multiple *Data Nodes*, describing what is measured from the device it belongs to (e.g. temperature, engine RPM).

2.3 General information

In the examples that will follow, *server-url* stands for a URL provided for the WRM instance to which the API calls are to be made and the version number.

> ⓘ server-url = {base-url}/api/v{version-number}
> By default the complete API URL will be https://my.iot-ticket.com/api/v1.

2.4 Communication Security

All requests to the API are made through HTTPS. With HTTPS, the HTTP protocol is protected from wiretapping and man-in-the-middle attacks, therefore data being transferred is secure. Authentication to the API will be via HTTP Basic with username and password pair and should be sent along all requests that require it. Below is an example of a request using command line program *curl*. The user name and password are supplied with the –*user* option.

```
curl -X POST --user wrmuser:wrmpassword \
     -H "Content-Type:application/json" \
     -d '{"name":"OPC Server",
         "manufacturer":"HP",
         "type":"PC",
         "attributes":[{"key":"OS","value":"Windows 7"},
                    {"key":"Screen-size","value":"1960*1800"}]
       }' \
"https://{server-url}/devices"
```

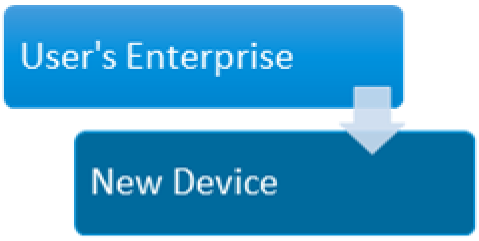
2.5 Quota Policy

WRM limits the number of requests per day, amount of data that can be stored, number of devices and number of data nodes per device. Each of these quotas is enforced on a per-client basis. Users should visit the WRM webpage on how to increase allocated quotas.

Subject to change, free users are currently restricted to five devices, a maximum of 20 data nodes per device and a total of 20,000 read request per day per device. The total amount of data storage for a free user is also restricted to 50 Megabytes. When the quota is exceeded, an HTTP Response code 403 with the specific error code is sent in the response. For more details, see the Error Handling chapter below. The read request quotas are reset at midnight, UTC time.

3 Device Management

A registered user by default has an enterprise root object. When a device is registered with a client's credential it immediately shows up under the client's enterprise.

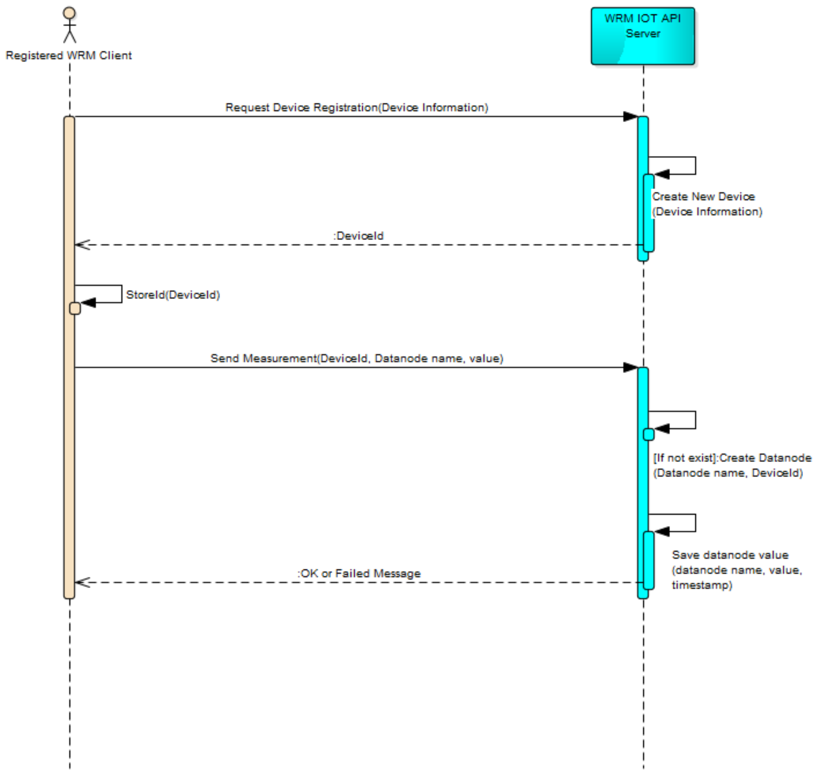


4 Device hierarchy

A device ID is automatically assigned to a newly registered device; the API Client should store this device ID as it uniquely identifies the device and will be used in subsequent calls.

> Client should avoid multiple registration call as this might result to duplicate devices being created.

When in doubt, a good flow will be to get the list of already created devices and validate the device's existence on the server through its name and attributes. Once the device is registered and the device ID is obtained, clients can immediately start sending measurement values to the server. The figure below illustrates this sequence.


Registered WRM Client
WRM IOT API Server
Request Device Registration(Device Information)
Create New Device (Device Information)
:DeviceId
StoreId(DeviceId)
Send Measurement(DeviceId, Datanode name, value)
[If not exist]:Create Datanode (Datanode name, DeviceId)
Save datanode value (datanode name, value, timestamp)
:OK or Failed Message

5 Create device flow

3.1 Register a device

URL: /devices/

HTTP method: POST

This call will create a new device under the authenticated client's enterprise unless the enterpriseId is specified.

Parameter	Description	Example

Object	A Json object with the following fields:
	• Name (required *)
	• Manufacturer (required)
	• Type
	• Description
	• Attributes
	• EnterpriseId
	• getOrCreateFilter
		{
		  "name": "OPC Server",
		  "manufacturer": "HP",
		  "type": "PC",
		  "description": "The main server for process data",
		  "attributes": [
		    {
		      "key": "Application Version",
		      "value": "0.2.3"
		    },
		    {
		      "key": "Chip",
		      "value": "Corei7"
		    }
		  ],
		  "enterpriseId": "E50"
		}

Name (String:100) (required *)	A short name for the device. It is visible in the enterprise tree when the client logs in to the WRM Desktop.	NX100, iPhone 5
Manufacturer (String:100)	Short name for the device's manufacturer.	ABB, Apple
Type (String:100)	This field should describe the main category the device belongs to.	PC, server, mobile phone, sensor, vehicle
Description (String:255)	A description of the device: what it does or where it is located.	Frequency converter for the ship's engine.
Attributes (key, String:255) (value, String:255)	Contains arrays of key value pairs. This is used to store additional attributes for the devices as desired by the client. A maximum of 50 attributes is accepted.	Additional attributes for an AC Drive may include: Inputphase : Three phases InputVoltage: 380-420V Options: EMC2, QPES

Parameter	Description	Example
enterpriseId	Enterprise Id under which the device should be created. If Enterprise Id is not given (or left empty) then the device will be created under enterprise that current user belongs to. Enterprise Id is given as a string and server accepts the Id with and without the prefix "E".	50, E50

getOrCreateFilter	Filter for getting device and creating it only if it doesn't already exist. If device matching the filter is found, it is not created but only returned as a response to this request.
	
	name or attributes or any combination of those are required. This allows to filter only by name or attributes and to allow duplicate entries on any of those if desired. If name is not provided here it needs to be provided as regular parameter. Values in filter overwrite regular name and attributes if they are set. Search for name is case insensitive and search for attributes is case sensitive.
	
	Searches only devices directly one level under the user's enterprise/ provided enterprise.
		```
		{
		  "name": "Device 1",
		  "attributes": [
		    {
		      "key": "Application Version",
		      "value": "0.2.3"
		    },
		    {
		      "key": "Chip",
		      "value": "Corei7"
		    }
		  ]
		}
		```

** Required if not present in getOrCreateFilter.*

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details, see the Error Handling chapter below.

Response to register a device

Field	Description
href	The URL to access the resource.
deviceID	The ID of the device, to be used in subsequent calls to write and read the device data nodes. It consists of 32 alphanumeric characters.
name	The name provided for the device when registering it.
manufacturer	The manufacturer name provided when registering it.
type	The type provided for the device when registering it.
createdAt	The time the device was created on the server. The ISO8601 format (UTC) format is used.
attributes	The device attributes. Name value pair provided when registering the device.
description	Description of the device if any.
enterpriseId	Enterprise Id of the Enterprise the device belongs to.
enterpriseName	Name of the Enterprise the device belongs to.
resourceId	The resource id for the device.
created	Boolean value if a new device was registered (only present when getOrCreateFilter is used).

Example

Request
https://\{server-url}/devices/ **REQUEST PAYLOAD is as shown in the example above.**

JSON Response

```
{
   "attributes": [{ "key": "Application version", "value": "0.2.3" },
         { "key": "Chip", "value": "Corei7" }],
   "createdAt": "2014-12-03T10:31:05UTC",
   "deviceId": "153ffceb982745e8b1e8abacf9c217f3",
   "href": "https://{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3",
   "name": "OPC Server",
   "manufacturer": "HP",
   "type": "PC",
   "enterpriseId": "E50",
   "enterpriseName": "Enterprise 1",
   "resourceId": "X123"
}
```

3.2 Get devices

This call returns a list consisting of general information about the devices the client has access to.

URL: /devices

HTTP method: GET

Authentication Required: Yes

Parameter	Description	Example
callback	JavaScript function to be executed when the response is received (JSONP).	/devices?callback=foo returns: foo(response data);
format	The format: json or xml; the default format is json.	/devices?format=xml
limit (integer)	The number of results to return. Maximum value is 100, it defaults to 10 when no value is provided.	/devices?limit=5
offset (integer)	The number of results to skip from the beginning.	/devices?offset=3 Paging example: /devices?offset=0&limit=100 /devices?offset=100&limit=100
hasDatanode	It gets only the devices that contain the datanode. If the value starts with a slash character, then the path is also checked in query; if not, then only the datanode name is checked.	/devices?hasDatanode=/path/to/temp1, /path/to/temp2, temp3
hasMetadata	It gets only the devices that contain the metadata.	/devices?hasMetadata=metadata1:value1, metadata2:value2
enterpriseId	It gets only devices that are under the enterprise with enterpriseId. Enterprise id can be provided with or without the "E" character in front of the id.	/devices?enterpriseId=1234

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details, see the Error Handling chapter below.

Response to get devices

Field	Description
offset	The number of results skipped from the beginning.
limit	The amount of results per page.
fullSize	The total number of devices that the client has access to.

Field	Description
devices	The list of device objects. Each of the object contains: • href • name • manufacturer • type (if any) • createdAt • description • attributes • enterpriseId • enterpriseName • resourceId
href	The URL to the device.
name	The name of the device provided when registering it.
manufacturer	The manufacturer name provided when registering the device.
type	The type provided for the device when registering it.
createdAt	The time the device was registered using ISO8601 format.
attributes	The device attributes. Name value pair provided when registering the device.
description	Description of the device if any.
enterpriseId	Enterprise Id of Enterprise that the device belongs to.
enterpriseName	Name of the enterprise that the device belongs to.
resourceId	The resource id of the device.

Example

Request
[**https://\{server-url}/devices** **REQUEST PAYLOAD:NONE**
JSON Response

```
{
   "fullSize": 28,
   "limit": 10,
   "offset": 0,
   "items": [
      {
         "attributes": [{ "key": "OS", "value": "Windows 7" },
        { "key": "Screen Size", "value": "30 Inches" }],
   "createdAt": "2014-12-03T10:31:05UTC",
   "deviceId": "153ffceb982745e8b1e8abacf9c217f3",
   "href": "https://{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3",
         "name": "OPC Server",
         "manufacturer": "HP",
   "type": "PC",
         "enterpriseId": "E50",
         "enterpriseName": "Enterprise 1",
         "resourceId": "X123"
      },
      {
         "attributes": [],
         "createdAt": "2014-12-03T08:55:14UTC",
         "deviceId": "e057aba9cad84a3aa3fc6b99bbe2196e",
         "href": "https://{server-url}/devices/e057aba9cad84a3aa3fc6b99bbe2196e",
         "name": "GT-I9295",
         "manufacturer": "Samsung",
         "type": "MobilePhone",
         "entepriseId": "E50",
         "enterpriseName": "Enterprise 1",
         "resourceId": "X124"
      },
      ....
   ]
}
```

XML Response

```xml
<devicesResult>
  <fullSize>28</fullSize>
  <limit>10</limit>
  <offset>0</offset>
  <items>
    <device>
      <attributes />
      <createdAt>2014-12-10T10:40:17UTC</createdAt>
      <description>The main server for process data</description>
      <deviceId>e057aba9cad84a3aa3fc6b99bbe2196e</deviceId>
      <enterpriseId>E50</enterpriseId>
      <enterpriseName>Enterprise 1</enterpriseName>
      <href>https://{server-url}/devices/e057aba9cad84a3aa3fc6b99bbe2196e</href>
      <name>GT-19295</name>
      <resourceId>X123</resourceId>
      <type>MobilePhone</type>
      <manufacturer>Samsung</manufacturer>
    </device>
    <device>
      <attributes>
        <attribute key="Application Version" value="0.2.3"/>
        <attribute key="Chip" value="Corei7"/>
      </attributes>
      <createdAt>2014-12-10T10:44:03UTC</createdAt>
      <description>The main server for process data</description>
      <deviceId>153ffceb982745e8b1e8abacf9c217f3</deviceId>
      <enterpriseId>E50</enterpriseId>
      <enterpriseName>Enterprise 1</enterpriseName>
      <href>https://{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3</href>
      <name>OPC Server</name>
      <resourceId>X124</resourceId>
      <type>PC</type>
      <manufacturer>HP</manufacturer>
    </device>
  <!--more devices -->
  </items>
</devicesResult>
```

3.3 Get a single device

URL: /devices/deviceId
HTTP method: GET
Authentication Required: Yes
This call gets general information for the device with the provided ID.

Parameter	Description	Example
callback	JavaScript function to be executed when the response is received (JSONP).	/devices/deviceId?callback=foo returns: foo(response data);

Parameter	Description	Example
format	The format: json or xml; the default format is json.	/devices/ deviceId?format=xml

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object with more information. For more details, see the Error Handling chapter below.

Response to get a device

Field	Description
href	The URL to the device.
deviceId	The ID of the device, to be used in subsequent calls to write and read the device data nodes. It consists of 32 alphanumeric characters.
name	The name of the device provided when registering it.
manufacturer	The manufacturer name provided when registering the device.
type	The type provided for the device when registering it.
createdAt	The time the device was registered using ISO8601 format.
attributes	The device attributes. Name value pair provided when registering the device.
description	Description of the device if any.
enterpriseId	Enterprise Id of Enterprise that device belongs to.
enterpriseName	Name of the enterprise that the device belongs to.
resourceId	The resource id of the device.

Example

GET Request
[**https://\{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3**   **REQUEST PAYLOAD:NONE**
JSON Response

```
{
    "attributes": [],
    "createdAt": "2014-12-03T08:55:14UTC",
    "deviceId": "e057aba9cad84a3aa3fc6b99bbe2196e",
    "href": "https://{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3",
    "name": "GT-I9295",
    "description": "The main server for process data",
    "type": "MobilePhone",
    "manufacturer": "Samsung",
    "enterpriseId": "E50",
    "enterpriseName": "Enterprise 1",
    "resourceId": "X123"
}
```

XML Response

```
<device>
  <attributes>
    <attribute key="Application Version" value="0.2.3"/>
    <attribute key="Chip" value="Corei7"/>
  </attributes>
  <createdAt>2014-12-03T10:44:03UTC</createdAt>
  <description>The main server for process data</description>
  <deviceId>153ffceb982745e8b1e8abacf9c217f3</deviceId>
  <enterpriseId>E50</enterpriseId>
  <enterpriseName>Enterprise 1</enterpriseName>
  <href>https://{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3</href>
  <name>OPC Server</name>
  <resourceId>X123</resourceId>
  <type>PC</type>
  <manufacturer>HP</manufacturer>
</device>
```

3.4 Get a device's datanode list

URL: /devices/deviceId/datanodes/

HTTP method: GET

Authentication Required: Yes

This call gets a list of provided device datanodes.

Parameter	Description	Example
callback	JavaScript function to be executed when the response is received (JSONP).	/devices/deviceId/datanodes?callback=foo returns: foo(response data);
limit (integer)	The number of results to return. Maximum value is 100, it defaults to 10 when no value is provided.	/devices/deviceId/datanodes?limit=5

Parameter	Description	Example
offset (integer)	The number of results to skip from the beginning.	/devices/deviceId/datanodes?offset=3
format	The format: json or xml; the default format is json.	/devices/deviceId/datanodes?format=xml

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details, see the Error Handling chapter below.

Response to get a device's datanodes

Field	Description
offset	The number of results skipped from the beginning.
limit	The amount of results per page.
fullSize	The total number of hits to the query.
items	The list of datanode objects.

Example

GET Request
[**https://\{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3/datanodes** **REQUEST PAYLOAD:NONE**
JSON Response

```
{
  "fullSize": 2,
  "limit": 10,
  "offset": 0,
  "items": [
    {
      "unit": "c",
      "dataType": "double",
      "href": "https://{server-url}/process/read/4ec1b0221f794f0f990e419bcc9a15cf?
datanodes=Engine/Core/Temperature",
      "name": "Temperature",
      "path": "Engine/Core"
    },
    {
      "unit": "Hz",
      "dataType": "long",
      "href": "https://{server-url}/process/read/4ec1b0221f794f0f990e419bcc9a15cf?
datanodes=Cycles",
      "name": "Cycles",

    }]
}
```

XML Response

```
<datanodeResult>
  <fullSize>2</fullSize>
  <limit>10</limit>
  <offset>0</offset>
  <items>
    <datanode unit="c">
      <dataType>double</dataType>
      <href>https://{server-url}/process/read/4ec1b0221f794f0f990e419bcc9a15cf?datanodes=Engine/
Core/Temperature</href>
      <name>Temperature</name>
      <path>Engine/Core</path>
    </datanode>
    <datanode unit="Hz">
      <dataType>long</dataType>
      <href>https://{server-url}/process/read/4ec1b0221f794f0f990e419bcc9a15cf?datanodes=Cycles</
href>
      <name>Cycles</name>
    </datanode>
  </items>
</datanodeResult>
```

3.5 Get a device's virtual data tag list

URL: /devices/deviceId/datanodes/virtual

HTTP method: GET

Authentication Required: Yes

Get a list of provided device virtual data tags

Parameter	Description	Example
callback	JavaScript function to be executed when the response is received (JSONP)	/devices/deviceId/datanodes/virtual?callback=foo returns: foo(response data);
limit (integer)	Number of results to return. Maximum of 100, defaults to 10 when none is provided	/devices/deviceId/datanodes/virtual?limit=5
offset (integer)	Number of result to skip from the beginning.	/devices/deviceId/datanodes/virtual?offset=3
format	Format (json or xml) default is json	/devices/deviceId/datanodes/virtual?format=xml

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see Error Handling chapter.

Response to get device virtual data tags

Field	Description
offset	The number of results skipped from the beginning.
limit	The amount of results per page.
fullSize	The total number of hits to the query.
items	The list of datanode objects.

Example

GET Request
[**https://\{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3/datanodes/virtual** **REQUEST PAYLOAD:NONE**
JSON Response

```
{
    "fullSize": 2,
    "limit": 10,
    "offset": 0,
    "items": [
        {
            "dataType": "double",
            "href": "https://{server-url}/process/read/4ec1b0221f794f0f990e419bcc9a15cf?
vtags=Temperature",
            "name": "Temperature"
        },
        {
            "dataType": "long",
            "href": "https://{server-url}/process/read/4ec1b0221f794f0f990e419bcc9a15cf?vtags=Cycles",
            "name": "Cycles"
        }]
}
```

XML Response

```
<datanodeResult>
  <fullSize>2</fullSize>
  <limit>10</limit>
  <offset>0</offset>
  <items>
    <datanode
      <dataType>double</dataType>
      <href>https://{server-url}/process/read/4ec1b0221f794f0f990e419bcc9a15cf?vtags=
Temperature</href>
      <name>Temperature</name>
    </datanode>
    <datanode>
      <dataType>long</dataType>
      <href>https://{server-url}/process/read/4ec1b0221f794f0f990e419bcc9a15cf?vtags=Cycles</href>
      <name>Cycles</name>
    </datanode>
  </items>
</datanodeResult>
```

3.6 Get data tag and device information with MID value

URL: /datatags/mids

HTTP method: GET

Authentication Required: Yes

Get information about the device and the data tag fetched via data tag's MID value. Only the External device's non-virtual data tags are supported. The query will ignore MIDs, which aren't visible to the user and aren't the supported type. The query takes following parameters:

Parameter	Description	Example
mids	Required parameter which defines data tag(s) that should be fetched. Only the External device's non-virtual data tags to which has access to are fetchable.	/datatags/mids?mids=303,652,653,742,845
format	Optional parameter for defining the format (json or xml) of request's response, default is json	/datatags/mids?mids=385&format=json

The query returns following information about the matching MIDs:

Field	Description
MID	The MID value of the requested data tag.
deviceId	Device ID information of the device to which the request data tag is related to.
path	The location path from the data tag's device to the data tag's location. If the data tag is located directly at the device, then the path value is empty ("").
tagName	The name of the requested data tag.

Example:

GET Request
[**https://\{server-url}/datatags/mids?mids=324,325,350,360**   **REQUEST PAYLOAD:NONE**
JSON Response

```
{
  "tagInfoList": [
    {
      "MID": 324,
      "deviceId": "abcdefg123",
      "path": "",
      "tagName": "TimerTag"
    },
    {
      "MID": 325,
      "deviceId": "abcdefg123",
      "path": "asset1",
      "tagName": "RESTValueTag"
    },
    {
      "MID": 360,
      "deviceId": "ed12efg56",
      "path": "collection/device1A",
      "tagName": "Sensor1A"
    }
  ]
}
```

XML Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataTagInfoList>
   <tagInfoList>
      <deviceId>abcdefg123</deviceId>
      <MID>324</MID>
      <path></path>
      <tagName>TimerTag</tagName>
   </tagInfoList>
   <tagInfoList>
      <deviceId>abcdefg123</deviceId>
      <MID>325</MID>
      <path>asset1</path>
      <tagName>RESTValueTag</tagName>
   </tagInfoList>
   <tagInfoList>
      <deviceId>ed12efg56</deviceId>
      <MID>360</MID>
      <path>collection/device1A</path>
      <tagName>Sensor1A</tagName>
   </tagInfoList>
</dataTagInfoList>
```

3.7 Move device

URL: /devices/move/deviceId/
HTTP method: POST
Authentication Required: Yes
Move device from current enterprise to another enterprise.

Parameter	Description	Example
Object	A Json (or xml) object with the following fields: • enterpriseId (required)	`{ "enterpriseId": "E50" }`
format	Format (json or xml), default is json	/devices/move/deviceId?format=xml
enterpriseId (String)	Enterprise Id from IoT-TICKET server under which to move the device. If the Enterprise Id matches to the Enterprise that the device already is in, the API returns also succesfully but does nothing for the device. Enterprise Id is given as a string and server accepts the Id with and without the prefix "E".	E50, 50

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see Error Handling chapter.

Response to move device

Field	Description
href	URL to the device
name	Name of the device provided when registering it.
type	Type of the device provided when registering it.
manufacturer	Manufacturer name provided when registering.
createdAt	The time the device was registered using ISO8601 format.
description	Description of the device provided.
attributes	The device attributes. Name value pair provided when registering the device.
enterpriseId	Enterprise Id of Enterprise that device belongs to

Example

POST Request

[**https://\{server-url}/devices/move/153ffceb982745e8b1e8abacf9c217f3**
REQUEST PAYLOAD: enterpriseId in JSON (or XML) format as shown in the example above

JSON Response

```
{
    "attributes": [],
    "createdAt": "2014-12-03T08:55:14UTC",
    "deviceId": "153ffceb982745e8b1e8abacf9c217f3",
    "href": "https://{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3",
    "name": "GT-I9295",
    "description": "The main server for process data",
    "type": "MobilePhone",
    "manufacturer": "Samsung",
    "enterpriseId": "E50"
}
```

XML Response

```
<device>
  <attributes>
    <attribute key="Application Version" value="0.2.3"/>
    <attribute key="Chip" value="Corei7"/>
  </attributes>
  <createdAt>2014-12-03T10:44:03UTC</createdAt>
  <description>The main server for process data</description>
  <deviceId>153ffceb982745e8b1e8abacf9c217f3</deviceId>
  <enterpriseId>E50</enterpriseId>
  <href>https://{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3</href>
  <name>OPC Server</name>
  <type>PC</type>
  <manufacturer>HP</manufacturer>
</device>
```

3.8 Delete device

URL: /devices/deviceId/

HTTP method: DELETE

Authentication Required: Yes

Delete device

Example

DELETE Request
[**https://\{server-url}/devices/153ffceb982745e8b1e8abacf9c217f3**   **REQUEST PAYLOAD: NONE**
JSON Response
NONE
XML Response
NONE

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see Error Handling chapter.

4 Reading and writing data

Data values are written to the device's datanodes. Each datanode is identified by its name and the path specified by the client. The datanode is created the first time it is encountered by the server. Intermediate nodes are also created if a path is specified the first time the datanode is encountered. The full path to the datanode should be specified when the datanode is to be read from.

Data types

1 Data types

Datanote data type	Type field in response	Format
Double (64bits)	double	Numerical value
Long (2^{63}-1 to -2^{63}) 64bits	long	Numerical value
String	string	Any Unicode character except " or \ which are escaped by |
Boolean	boolean	"true" or "false" without quotation marks
Binary	binary	Base64 encoded

4.1 Writing data

URL: /process/write/deviceId/
HTTP method: POST
Authentication Required: Yes
The user must write the provided values to the corresponding data node. A new data node is created when the server has not received a value prior to this, with the same name and path, if any. Subsequent writings always include the same name and path to write to the same data node.

Parameter	Description	Example
deviceId	The target device ID.	
Data to be written	The arrays of data object to be saved. Each of the objects has the following attributes: • name (required) • path (optional) • v- value (required) • ts –unix timestamp (optional) • unit (optional) <ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="b26feff9-595c-4882-8764-f4c2a435b4d9"><ac:plain-text-body><![CDATA[* dataType (optional)	[{ "name": "Temperature", "path": "MainEngine/Core", "v": 60, "ts": 1414488510057, "unit": "c"}, {"name":"Latitude","v":63, "dataType":" long" }, {"name":"Latitude","v": 65}, {"name": "Latitude","v": 67}]

Parameter	Description	Example
Name (string:100)	A short description of the datanode. A device's datanode is uniquely identified with its name and path. The value is case insensitive.	
Path: (String:1000)	• Forward slash separates the list of paths to be created to get to the datanode. The path can only consist of a maximum of 10 components. A slash at the beginning is simply ignored. For example, the paths /Engine/Cabin and Engine/Cabin are equivalent to each other. • When viewed from the WRM Desktop UI, intermediate nodes as specified by the path are created between the devices and the datanode in a nested tree structure. • Each component of the path can only contain alphanumeric values A-Za-z0-9. The value is case insensitive. • An empty path or a missing path attributes are equivalent to each other.	Engine/Cabin/ Example of an invalid path: 1/2/3/4/5/6/7/8/9/10/11
v	The value to be written. This must be applicable to the datatype, if provided.	
unit (String:10)	The unit associated with the data, preferably 1 or 2 characters.	c,Hz
ts (long)	Unix Timestamp. The number of milliseconds since the Epoch. When a timestamp is missing, the current timestamp is automatically used.	1414488510057
dataType	When the datatype is not provided, the possible data type is inferred from the value first received by the server.	Possible values are: long, double, boolean, string or binary. The value is case insensitive.

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details, see the Error Handling chapter below.

Notice: A single request payload can be at maximum 200 kilobytes. Bigger requests will result to an error code of 8000.

Response to write to the datanodes of a device

Field	Description
writeresult object	An array of response objects, each containing: • href • writtenCount
totalWritten	The total number of data points written.
href	The URL to read from the datanode targeted in the write.
writtenCount	The number of values written to that particular datanode.

Example

POST Request
[https://\{server-url}/process/write/153ffceb982745e8b1e8abacf9c217f3 **REQUEST PAYLOAD as shown in example above.**
JSON Response
see block below
XML Response
see block below

```
{"writeResults":
  [{
   "href": "https://{server-url}/process/read/153ffceb982745e8b1e8abacf9c217f3/?
datanodes=MainEngine/Core/Temperature",
        "writtenCount": 1
     },
   {
    "href": "https://{server-url}/process/read/153ffceb982745e8b1e8abacf9c217f3/?datanodes=/
Latitude",
        "writtenCount": 3
      }
    ],
    "totalWritten": 4
}
```

```
<writeResults>
    <writeResult>
        <href https://{server-url}/process/read/153ffceb982745e8b1e8abacf9c217f3/?
datanodes=MainEngine/Core/Temperature </href>
        <writtenCount>1</writtenCount>
    </writeResult>
    <writeResult>
        <href> https://{server-url}/process/read/153ffceb982745e8b1e8abacf9c217f3/?datanodes=/
Latitude </href>
        <writtenCount>3</writtenCount>
    </writeResult>
    <totalWritten>4</totalWritten>
</writeResults>
```

4.2 Reading data

URL: /process/read/deviceId/
HTTP method: GET
Authentication Required: Yes
This call reads device datanode and/or virtual data tag values.

Parameter	Description	Example
format	The format: json or xml; the default format is json.	/process/read/deviceId?datanodes=latitude&format=xml
deviceId	The target device ID.	
datanodes	Comma separated datanode names or full paths to be read from. If the name provided matches more than one datanode, they are all included in the response. The number of datanodes in the path should not exceed 10. If there are multiple datanodes with the same name, include the path to be more specific. For example, two datanodes named "Temp", one having the path "Engine/core" and the second one having no path. The specific read path will be ?datanode=/Engine/coreTemp and ?datanodes=/Temp, respectively. The read path ?datanodes=Temp will return results for both datanodes.	/process/read/deviceId?datanodes =latitude,longitude,altitude
vtags	Comma separated virtual data tag names to be read from. The number of virtual data tags and data nodes in total should not exceed 10.	/process/read/deviceId?vtags=virtualtag1,virtualtag2
fromdate (long)	Unix Timestamp. The number of milliseconds since the Epoch. It defines the time from which the data is obtained. It should be provided, if there is a todate.	/process/read/deviceId?datanodes=longitude&fromdate=1415260152284
todate (long)	Unix Timestamp. The number of milliseconds since the Epoch. It defines the time to which the data read ends. It defaults to the current time if this value is not not provided and a fromdate exists. If neither the fromdate and todate are provided, the latest value is returned.	/process/read/deviceId?datanodes=longitude&fromdate=1415260152284&todate=1417609677
limit (integer)	The maximum number of data points returned for each datanode queried. It defaults to 1000 if not provided and has a maximum value of 10,000.	/process/read/deviceId?datanodes=longitude&limit=3
order	It orders the values by timestamp, in either ascending or descending order. The possible values are *ascending* and *descending*. The default is ascending.	/process/read/deviceId?datanodes=longitude&order=descending

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details, see the Error Handling chapter below.

Response to read the datanodes of a device

Field	Description
readresult Object	The array of objects contain: • name • type • unit • value object containing arrays of v and ts fields
name	The name of the datanode.
path	The path to the datanode, if any.

Field	Description
v	The value at timestamp ts.
ts	The timestamp associated with the value. This is always added even if the client did not include a timestamp at the time of writing.
unit	The unit associated with the data node.

Example

GET Request
[https://\{server-url}/process/read/153ffceb982745e8b1e8abacf9c217f3?datanodes=MainEngine/Core/Temperature,Latitude&fromdate=1417636256406&limit=3 **REQUEST PAYLOAD: NONE**
JSON Response

```
{
    " datanodeReads": [{
        "name": "Latitude",
        "dataType": "long",
        "values": [{
            "v": "60","ts": 1417636260139}]
    },
    {
        "name": "Temperature",
        "path": "Engine/Core",
        "unit": "c",
        "dataType": "double",
        "values": [
        {"v": "65","ts": 1417636260152},
        {"v": "63","ts": 1417636260152},
        {"v": "73","ts": 1417636260152}]
    }],
    "href":"https://{server-url}/process/read/153ffceb982745e8b1e8abacf9c217f3?
datanodes=MainEngine/Core/Temperature,Latitude&fromdate=1417636256406&limit=3"
}
```

XML Response

```
<readResults>
  <href>https://{server-url}/process/read/153ffceb982745e8b1e8abacf9c217f3?
datanodes=MainEngine/Core/Temperature,Latitude&fromdate=1417636256406&limit=3</href>
  <datanodeReads dataType="long">
    <name>Latitude</name>
    <values>
      <value>
        <v>60</v>
        <ts>1417636260139</ts>
      </value>
    </values>
  </datanoderead>
  <datanoderead unit="c" dataType="double">
    <name>Temperature</name>
    <path>Engine/Core</path>
    <values>
      <value>
        <v>65</v>
        <ts>1417636260152</ts>
      </value>
      <value>
        <v>67</v>
        <ts>1417636260152</ts>
      </value>
      <value>
        <v>73</v>
        <ts>1417636260152</ts>
      </value>
    </values>
 </datanoderead>
</readResults>
```

4.3 Reading statistical data

URL: /stat/read/deviceId/

HTTP method: GET

Authentication Required: Yes

Read statistical data node and/or virtual data tag values from a device.

Parameter	Description	Example
format	Format (json, xml or csv) default is json.	/stat/read/deviceId?datanodes=longitude&fromdate=1415260152284&todate=1417609677000&grouping=hour&format=xml
deviceId	The target device id.	

Parameter	Description	Example
datanodes	Comma separated data node names or fullpaths to be read from. If the name provided matches more than one datanode, they are all included in the response. The number of datanodes in the path should not exceed 10. If there are multiple data node with the same name, include the path to be more specific. For example two datanodes with name "Temp" with path "Engine/core" and with no path. The specific read path will be ?datanode=/Engine/coreTemp and ?datanodes=/Temp respectively. The read path ?datanodes=Temp will return results for both datanodes.	/stat/read/deviceId?datanodes=longitude&fromdate=1415260152284&todate=1417609677000&grouping=hour
vtags	Comma separated virtual data tag names to be read from. The number of virtual data tags and data nodes in total should not exceed 10.	/stat/read/deviceId?vtags=virtualtag1,virtualtag2&fromdate=1415260152284&todate=1417609677000&grouping=hour
fromdate (long)	Unix Timestamp. Number of milliseconds since the Epoch. Defines the time from which the data is obtained. Must be provided. The time range defined by fromdate and todate is extended automatically to contain at least one unit of grouping interval.	/stat/read/deviceId?datanodes=longitude&fromdate=1415260152284&todate=1417609677000&grouping=hour
todate (long)	Unix Timestamp. Number of milliseconds since the Epoch. Defines the time to which the data read ends. Must be provided. The time range defined by fromdate and todate is extended automatically to contain at least one unit of grouping interval.	/stat/read/deviceId?datanodes=longitude&fromdate=1415260152284&todate=1417609677000&grouping=hour
order	Orders the values by timestamp either in ascending or descending order. Possible values are *ascending* and *descending*. **Default is ascending.**	/stat/read/deviceId?datanodes=longitude&fromdate=1415260152284&todate=1417609677000&grouping=hour&order=descending
grouping	Determines the grouping type for the statistical data. Possible values are: "minute", "hour", "day", "week", "month","year". Must be provided.	/stat/read/deviceId?datanodes=longitude&fromdate=1415260152284&todate=1417609677000&grouping=hour

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error Handling chapter.

Response to read statistical data from device's data nodes

Field	Description
datanodeReads	Array of objects containing • name • dataType • unit • path • values
name	The name of the data node
datatype	The data type of the data node.

Field	Description
unit	The unit associated with the data node.
path	The path to the data node if any.
values	Array of value objects containing • min (left out if interval had no values) • max (left out if interval had no values) • avg (left out if interval had no values) • count • sum • ts
min	The minimum value of the data node in the value object's time inteval.
max	The maximum value of the data node in the value object's time inteval.
avg	The average value of the data node in the value object's time inteval.
count	The total number of values for the data node in the value object's time inteval.
sum	The sum of values for the data node in the value object's time inteval.
ts	The start timestamp of the value object's time interval.

Example

GET Request
[**https://\{server-url}/stat/read/dUcYTaFfXD69gCSQYAVjS8?datanodes=Pressure&fromdate=1483228800000&todate=1550831791000&grouping=year** **REQUEST PAYLOAD: NONE**
JSON Response

```
{
   "href": "https://{server-url}/stat/read/dUcYTaFfXD69gCSQYAVjS8?
todate=1550831791000&grouping=year&datanodes=Pressure&fromdate=1483228800000",
   "datanodeReads": [
      {
         "dataType": "double",
         "unit": "bar",
         "name": "Pressure",
         "values": [
            {
               "count": 0,
               "sum": 0,
               "ts": 1483228800000
            },
            {
               "min": 34.5,
               "max": 65.7,
               "avg": 45.0,
               "count": 6546,
               "sum": 7674.6,
               "ts": 1514764800000
            },
            {
               "min": 32.5,
               "max": 67.8,
               "avg": 46.4,
               "count": 3423,
               "sum": 7564.2,
               "ts": 1546300800000
            }
         ]
      }
   ]
}
```

XML Response

```
<readResults>
   <href>https://{server-url}/stat/read/dUcYTaFfXD69gCSQYAVjS8?
todate=1550831791000&format=xml&grouping=year&datanodes=Pressure&fromdate=1483228800000
</href>
   <datanodeReads dataType="double" unit="bar">
      <name>Pressure</name>
      <values>
         <value>
            <count>0</count>
            <sum>0</sum>
            <ts>1483228800000</ts>
         </value>
         <value>
            <min>34.5</min>
            <max>65.7</max>
            <avg>45.0</avg>
            <count>6546</count>
            <sum>7674.6</sum>
            <ts>1514764800000</ts>
         </value>
         <value>
            <min>32.5</min>
            <max>67.8</max>
            <avg>46.4</avg>
            <count>3423</count>
            <sum>7564.2</sum>
            <ts>1546300800000</ts>
         </value>
      </values>
   </datanodeReads>
</readResults>
```

4.4 Datanode creation

URL: /process/create/deviceId/
HTTP method: POST
Authentication Required: Yes
Request creates a datanode with the given attributes.

Parameter	Description	Example
deviceId	The target device ID.	
Data to be written	An object with the following attributes: • name (required) • path (optional) • unit (optional) • dataType (required)	**{ "name": "Temperature", "path": "MainEngine/Core", "unit": "c", "dataType":"long" }**

Parameter	Description	Example
Name (string:100)	A short description of the datanode. A device's datanode is uniquely identified with its name and path. The value is case insensitive.	
Path: (String:1000)	• Forward slash separates the list of paths to be created to get to the datanode. The path can only consist of a maximum of 10 components. A slash at the beginning is simply ignored. For example, the paths /Engine/Cabin and Engine/ Cabin are equivalent to each other. • When viewed from the WRM Desktop UI, intermediate nodes as specified by the path are created between the devices and the datanode in a nested tree structure. • Each component of the path can only contain alphanumeric values A-Za-z0-9. The value is case insensitive. • An empty path or a missing path attributes are equivalent to each other.	Engine/Cabin/
Note: Example of an invalid path: 1/2/3/4/5/6/7/8/9/10/11		
unit (String:10)	The unit associated with the data, preferably 1 or 2 characters.	c,Hz
dataType	When creating a datatag the dataType must be provided.	Possible values are: long, double, boolean, string or binary.

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details, see the Error Handling chapter below.
Response to create datanode for a device

Field	Description
mid	Measurement id of the created datanode
name	Name of the datanode
unit	Unit of the datanode
dataType	Data type of the datanode
path	Path of the datanode

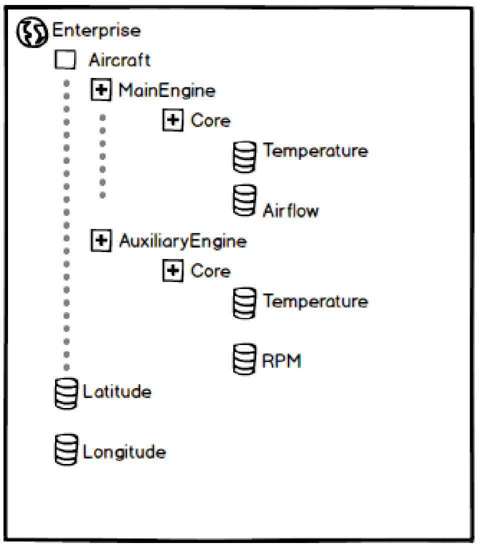
Example

POST Request
[**https://\{server-url}/process/create/153ffceb982745e8b1e8abacf9c217f3** **REQUEST PAYLOAD as shown in example above.**
JSON Response

```
{
"name": "Temperature",
"path": "MainEngine/Core",
"unit": "c",
"dataType":"long",
"mid": 100
}
```

4.5 Example of a device with hierarchical components

A device with hierarchical components can be easily modelled using the data node's path. Such a structure is illustrated here with a contrived example of an aircraft with a main engine and an auxiliary engine. The temperature and airflow in the main engine are to be measured, while the RPM and the temperature are measured in the auxiliary engine, at the same time. The aircraft also has a latitude and longitude measurement reading as shown in the figure below:



6 Example of a device with hierarchical components

First, the device is registered with the name "Aircraft" and manufacturer "Boeing", for example, after which a device ID is obtained and stored.
With the device ID, the main engine core temperature is written to a datanode named Temperature with a path *MainEngine/Core*. The auxiliary engine core temperature is written to a datanode named Temperature also, but with a path *Auxiliary/Core*, while the aircraft Latitude is named Latitude, with no path provided, as listed in the table below:

Measurement point	Name	Path
MainEngine Core Temperature	Temperature	MainEngine/Core
AuxiliaryEngine Core Temperature	Temperature	AuxiliaryEngine/Core

Measurement point	Name	Path
Latitude	Latitude	

Similarly, the Airflow and RPM are written with the same name but to path *MainEngine/Core* and to path *Auxiliary/Core*, respectively. To read a particular datanode's values, the full path to the datanode in the form **{PATH} /{NAME}** should be provided, therefore to read the main engine core temperature for example, the URL to use is:
/process/read/deviceId/?datanodes=MainEngine/Core/Temperature

To read the main engine core temperature, latitude, longitude and auxiliary RPM datanodes, the URL to use is:
/process/read/deviceId/?datanodes=MainEngine/Core/
Temperature,Latitude,Longitude,AuxiliaryEngine/Core/RPM

The datanodes could as well be named *MainEngineCoreTemperature* and *AuxillaryEngineCoreTemperature* for example, however this does not lean itself to aggregate queries and viewing the aircraft nested structure in the WRM Desktop UI. An example of an aggregated query will be to read all datanodes named temperature for the device. Using the URL / process/read/deviceId/?datanodes=Temperature, the server will return the values for both the Main Engine and Auxiliary Engine core **Temperature** data nodes.

ⓘ It should be noted that aggregated write request is not allowed, the path must always be specified if any.

5 Enterprise management

Enterprises can be managed by the enterprise API.
Get root enterprises

Returns a list consisting of the root enterprises the client has access to.
URL: /enterprises
HTTP method: GET
Authentication Required: Yes

Parameter	Description	Example
format	Format (json or xml) default is json	/enterprises?format=xml
limit (integer)	Number of results to return. Maximum of 100, defaults to 10 when none is provided	/enterprises?limit=5
offset (integer)	Number of result to skip from the beginning.	/enterprises?offset=3 Paging example: /enterprises?offset=0&limit=100 /enterprises?offset=100&limit=100

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error handling chapter.

Response to get root enterprises

Field	Description
offset	Number of results skipped from the beginning
limit	Amount of results per page
fullSize	The total number of root enterprises that the client has access to.
items	List of enterprise objects. Each of the objects contain • href • name • resourceId • hasSubEnterprises • attributes
href	URL to the enterprise
name	Name of the enterprise.
resourceId	The resource id of the enterprise.
hasSubEnterprises	Boolean (true/false) result if enterprise has sub enterprises.
attributes	Attributes for the enterprise (if exists)

Example

GET Request

[https://\{server-url}/enterprises
REQUEST PAYLOAD: NONE

JSON Response

```
{
   "fullSize": 28,
   "limit": 10,
   "offset": 0,
   "items": [
      {
         "href": "https://{server-url}/enterprises/E1234",
         "name": "Enterprise 1",
         "resourceId": "E1234",
         "hasSubEnterprises": true,
         "attributes": [{
            "key": "key1",
            "value": "value1"
         },
         {
            "key": "key2",
            "value": "value2"
         }]
      },
      {
         "href": "https://{server-url}/enterprises/E4567",
         "name": "Enterprise 2",
         "resourceId": "E4567",
         "hasSubEnterprises": false
      },
      ....
   ]
}
```

XML Response

```
<enterprisesResult>
  <items>
    <enterprise>
      <href>https://{server-url}/enterprises/E1234</href>
      <name>Enterprise 1</name>
      <resourceId>E1234</resourceId>
      <hasSubEnterprises>true</hasSubEnterprises>
      <attributes>
        <attribute key="key1" value="value1"/>
        <attribute key="key2" value="value2"/>
      </attributes>

    </enterprise>
    <enterprise>
      <href>https://{server-url}/enterprises/E4567</href>
      <name>Enterprise 2</name>
      <resourceId>E4567</resourceId>
      <hasSubEnterprises>false</hasSubEnterprises>
    </enterprise>
    <!-- more enterprises -->
  </items>
  <fullSize>28</fullSize>
  <limit>10</limit>
  <offset>0</offset>
</enterprisesResult>
```

5.1 Get enterprises under an enterprise

Returns a list of enterprises under the enterprise with the provided resource id. Enterprise's id can be provided either with or without the "E" prefix.

URL: /enterprises/enterpriseId

HTTP method: GET

Authentication Required: Yes

Parameter	Description	Example
format	Format (json or xml) default is json	/enterprises/enterpriseId?format=xml
limit (integer)	Number of results to return. Maximum of 100, defaults to 10 when none is provided	/enterprises/enterpriseId?limit=5
offset (integer)	Number of result to skip from the beginning.	/enterprises/enterpriseId?offset=3 Paging example: /enterprises/enterpriseId?offset=0&limit=100 /enterprises/enterpriseId?offset=100&limit=100

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error handling chapter.

Response to get enterprises under enterprise

Field	Description
offset	Number of results skipped from the beginning
limit	Amount of results per page
fullSize	The total number of root enterprises that the client has access to.
items	List of enterprise objects. Each of the objects contain • href • name • resourceId • hasSubEnterprises • attributes
href	URL to the enterprise
name	Name of the enterprise.
resourceId	The resource id of the enterprise.
hasSubEnterprises	Boolean (true/false) result if enterprise has sub enterprises.
attributes	Attributes for the enterprise (if exists)

Example

GET Request
[**https://\{server-url}/enterprises/1234** **REQUEST PAYLOAD: NONE**
JSON Response

```
{
  "fullSize": 15,
  "limit": 10,
  "offset": 0,
  "items": [
    {
      "href": "https://{server-url}/enterprises/E5678",
      "name": "Enterprise 3",
      "resourceId": "E5678",
      "hasSubEnterprises": true,
      "attributes": [{
        "key": "key1",
        "value": "value1"
      },
      {
        "key": "key2",
        "value": "value2"
      }]
    },
    {
      "href": "https://{server-url}/enterprises/E6789",
      "name": "Enterprise 4”,
      "resourceId": "E6789",
      "hasSubEnterprises": false
    },
    ....
  ]
}
```

XML Response

```
<enterprisesResult>
  <items>
    <enterprise>
      <href>https://{server-url}/enterprises/E5678</href>
      <name>Enterprise 3</name>
      <resourceId>E5678</resourceId>
      <hasSubEnterprises>true</hasSubEnterprises>
      <attributes>
        <attribute key="key1" value="value1"/>
        <attribute key="key2" value="value2"/>
      </attributes>
    </enterprise>
    <enterprise>
      <href>https://{server-url}/enterprises/E6789</href>
      <name>Enterprise 4</name>
      <resourceId>E6789</resourceId>
      <hasSubEnterprises>false</hasSubEnterprises>
    </enterprise>
    <!-- more enterprises -->
  </items>
  <fullSize>15</fullSize>
  <limit>10</limit>
  <offset>0</offset>
</enterprisesResult>
```

5.2 Create enterprise

Creates enterprise under an enterprise.
URL: /enterprises
HTTP method: POST
Authentication Required: Yes

Parameter	Description	Example
Object	A Json object with the following fields: • parentEnterpriseId (required) • name (required *) • attributes • getOrCreateFilter	`{ "name": "Enterprise 2", "parentEnterpriseId": "E123", "attributes":[ { "key":"key1", "value":"value1" } ] }`
parentEnterpriseId (required)	The parent enterprise id.	E123 or 123

Parameter	Description	Example
name (required *)	Name for the new enterprise	Enterprise 2
attributes	Attributes for the new enterprise	`[{ "key": "key1", "value": "value1" }, { "key": "key2", "value": "value2" }]`
getOrCreateFilter	Filter for getting enterprise and creating it only if it doesn't already exist. If enterprise matching the filter is found, it is not created but only returned as a response to this request. name or attributes or any combination of those are required. This allows to filter only by name or attributes and to allow duplicate entries on any of those if desired. If name is not provided here it needs to be provided as regular parameter. Values in filter overwrite regular name and attributes if they are set. Search for name is case insensitive and search for attributes is case sensitive. **NOTE:** Searches only enterprises directly one level under the provided parent enterprise.	`{ "name": "Enterprise 2", "attributes": [{ "key": "key1", "value": "value1" }, { "key": "key2", "value": "value2" }] }`

** Required if not present in getOrCreateFilter.*

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error handling chapter.

Response to create enterprise

Field	Description
attributes	Attributes for the enterprise (if exists)
created	Boolean value if a new enterprise was created (only present when getOrCreateFilter is used).
hasSubEnterprises	Boolean (true/false) result if enterprise has sub enterprises.
href	URL to the enterprise
name	Name of the enterprise.
resourceId	The resource id of the enterprise.

Example

POST Request
[https://\{server-url}/enterprises **REQUEST PAYLOAD is shown in the example above**
JSON Response
```
{
  "attributes": [
    {
      "key": "key1",
      "value": "value1"
    }
  ],
  "hasSubEnterprises": false,
  "href": "https://{server-url}/enterprises/E1234",
  "name": "Enterprise 2",
  "resourceId": "E1234"
}
```

6 Reading events

Active and inactive events and event history can be read with the events API.
Read active and inactive events

Returns list of active and inactive events that have not been acknowledged and are visible to the user. The result list paging is controlled by the limit and firstRow parameters. If events API is called without firstRow parameter the first page is returned. Next page can be queried using the response's lastRowId as the firstRow parameter for the next query.
URL: /events
HTTP method: GET
Authentication Required: Yes

Parameter	Description	Example
format	Format (json, xml or csv) default is json	/events?format=xml
limit (integer)	Number of results to return. Maximum of 10000, defaults to 1000 when none is provided	/events?limit=5
resourceIds	Comma separated list of resource ids to get the events from.	/events?resourceIds=X123,E567
firstRow	The uuid of the first event to start the events listing from (that event is not included in the response).	/events?firstRow=9c-7ffffe95e0aaf511-E4702
startTime	Unix Timestamp. Number of milliseconds since the Epoch. Defines the time from which the data is obtained. The time range defined by startTime and endTime. Default value is one year earlier than the current date.	/events?startTime=1555998429000
endTime	Unix Timestamp. Number of milliseconds since the Epoch. Defines the time from which the data is obtained. The time range defined by startTime and endTime. Default value is the current date.	/events?endTime=1555998429000

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error handling chapter.

Response to get events

Field	Description
lastRowId	The uuid of the last event in the response. This value can be used as the firstRow query parameter for a following call to continue the list from the current response's list.

items	List of event objects. Each of the objects contain • ackMode • description • eventId • eventState • eventUuid • group • metadata • resourceId • severity • source • startTime • type

Example

GET Request

[**https://\{server-url}/events**
REQUEST PAYLOAD: NONE

JSON Response

```
{
  "items": [
    {
      "ackMode": "ANY",
      "description": "test event",
      "eventId": 156,
      "eventState": "ACTIVE",
      "eventUuid": "9c-7ffffe95d067d839-E4702",
      "group": "test event group",
      "resourceId": "E4702",
      "severity": "NORMAL",
      "source": "Enterprise: Test Enterprise",
      "startTime": 1555576661958,
      "type": "WARNING"
    }
  ],
  "lastRowId": "9c-7ffffe95d067d839-E4702"
}
```

XML Response

```xml
<eventsResult>
  <items>
    <event>
      <ackMode>ANY</ackMode>
      <description>test event</description>
      <eventId>156</eventId>
      <eventState>ACTIVE</eventState>
      <eventUuid>9c-7ffffe95d067d839-E4702</eventUuid>
      <group>test event group</group>
      <metadata/>
      <resourceId>E4702</resourceId>
      <severity>NORMAL</severity>
      <source>Enterprise: Test Enterprise</source>
      <startTime>1555576661958</startTime>
      <type>WARNING</type>
    </event>
  </items>
  <lastRowId>9c-7ffffe95d067d839-E4702</lastRowId>
</eventsResult>
```

6.1 Read event history

Returns list of events that have been acknowledged and are visible to the user. The result list paging is controlled by the limit and firstRow parameters. If events history API is called without firstRow parameter the first page is returned. Next page can be queried using the response's lastRowId as the firstRow parameter for the next query.

URL: /events/history

HTTP method: GET

Authentication Required: Yes

Parameter	Description	Example
format	Format (json, xml or csv) default is json	/events/history?format=xml
limit (integer)	Number of results to return. Maximum of 10000, defaults to 1000 when none is provided	/events/history?limit=5
resourceIds	Comma separated list of resource ids to get the events from.	/events/history?resourceIds=X123,E567
firstRow	The uuid of the first event to start the events listing from (that event is not included in the response).	/events/history?firstRow=9c-7ffffe95e0aaf511-E4702
startTime	Unix Timestamp. Number of milliseconds since the Epoch. Defines the time from which the data is obtained. The time range defined by startTime and endTime. Default value is one year earlier than the current date.	/events/history?startTime=1555998429000

Parameter	Description	Example
endTime	Unix Timestamp. Number of milliseconds since the Epoch. Defines the time from which the data is obtained. The time range defined by startTime and endTime. Default value is the current date.	/events/history?endTime=1555998429000
acknowledge By	User id for the user that has acknowledged the events to filter the events.	/events/history?acknowledgeBy=TESTUSER
comment	Comment text or part of the comment text to filter the events.	/events/history?comment=test

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error handling chapter.

Response to get events history

Field	Description
lastRowId	The uuid of the last event in the response. This value can be used as the firstRow query parameter for a following call to continue the list from the current response's list.
items	List of event objects. Each of the objects contain • ackMode • acknowledgeBy • akcnowledgeTime • comment • description • endTime • eventId • eventState • eventUuid • group • metadata • resourceId • severity • source • startTime • type

Example

GET Request
[**https://\{server-url}/events/history** **REQUEST PAYLOAD: NONE**
JSON Response

```json
{
  "items": [
    {
      "ackMode": "ANY",
      "acknowledgeBy": "TESTUSER",
      "acknowledgeTime": 1555404837120,
      "comment": "test",
      "description": "test desc",
      "endTime": 1555393443075,
      "eventId": 158,
      "eventState": "INACTIVE",
      "eventUuid": "9e-7ffffe95db6690fe-E4702",
      "group": "test group",
      "resourceId": "E4702",
      "severity": "HIGH",
      "source": "Enterprise: Test Enterprise",
      "startTime": 1555392196353,
      "type": "ALARM"
    }
  ],
  "lastRowId": "9e-7ffffe95db6690fe-E4702"
}
```

XML Response

```xml
<eventsResult>
  <items>
    <event>
      <ackMode>ANY</ackMode>
      <acknowledgeBy>TESTUSER</acknowledgeBy>
      <acknowledgeTime>1555404837120</acknowledgeTime>
      <comment>test</comment>
      <description>test desc</description>
      <endTime>1555393443075</endTime>
      <eventId>158</eventId>
      <eventState>INACTIVE</eventState>
      <eventUuid>9e-7ffffe95db6690fe-E4702</eventUuid>
      <group>test group</group>
      <metadata/>
      <resourceId>E4702</resourceId>
      <severity>HIGH</severity>
      <source>Enterprise: Test Enterprise</source>
      <startTime>1555392196353</startTime>
      <type>ALARM</type>
    </event>
  </items>
  <lastRowId>9e-7ffffe95db6690fe-E4702</lastRowId>
</eventsResult>
```

7 Dashboard management

Dashboard API allows users to list and create dashboards.

7.1 List dashboards

Lists dashboards from resources.
URL: /resources/dashboards
HTTP method: GET
Authentication Required: Yes

Parameter	Description	Example
format	Format (json or xml) default is json	/resources/dashboards?format=xml
limit (integer)	Number of results to return. Maximum of 100, defaults to 100 when none is provided	/resources/dashboards?limit=5
offset (integer)	Number of results to skip from the beginning.	/resources/dashboards?offset=3 Paging example: /resources/dashboards?offset=0&limit=100 /resources/dashboards?offset=100&limit=100
resourceIds (required)	The ids of the resources to fetch the dashboards from.	/resources/dashboards?resourceIds=E123,X234

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error handling chapter.

Response to list dashboards

Field	Description
offset	Number of results skipped from the beginning
limit	Number of results per page
fullSize	The total number of dashboards found for the request.
items	List of dashboard objects. Each of the objects contain • dashboardId • dashboardTemplateId (if based on template) • dashboardTemplateName (if based on template) • href • name • resourceId
dashboardId	The id of the dashboard.
dashboardTemplateId	The id of the dashboard template the dashboard is based on.
dashboardTemplateName	The name of the dashboard template the dashboard is based on.

Field	Description
href	Link to the dashboard in the IoT-Ticket accessible by web browser.
name	The name of the dashboard.
resourceId	The resource id of the resource the dashboard belongs to.

Example

GET Request
[**https://\{server-url}/resources/dashboards?resourceIds=E123** **REQUEST PAYLOAD: NONE**
JSON Response
see below
XML Response

```
{
   "fullSize": 2,
   "limit": 100,
   "offset": 0,
   "items": [
      {
         "dashboardId": "931HxVmPnL8lVNFFjZ3vL7",
         "dashboardTemplateId": 1351,
         "dashboardTemplateName": "Template 1",
         "href": "https://{server-url} /Dashboard/#desktop/931HxVmPnL8lVNFFjZ3vL7/1",
         "name": "Dashboard 1",
         "resourceId": "E123"
      },
      {
         "dashboardId": "EmsF8TH6KrAx3UbYZxqqm5",
         "href": "https://{server-url}/Dashboard/#desktop/EmsF8TH6KrAx3UbYZxqqm5/1",
         "name": "Dashboard 2",
         "resourceId": "E123"
      }
   ]
}
```

```xml
<dashboardsResult>
   <items>
      <dashboard>
         <dashboardId>931HxVmPnL8lVNFFjZ3vL7</dashboardId>
         <dashboardTemplateId>1351</dashboardTemplateId>
         <dashboardTemplateName>Template 1</dashboardTemplateName>
<href>http://{server-url}/Dashboard/#desktop/931HxVmPnL8lVNFFjZ3vL7/1</href>
         <name>Dashboard 1</name>
         <resourceId>E123</resourceId>
      </dashboard>
      <dashboard>
         <dashboardId>EmsF8TH6KrAx3UbYZxqqm5</dashboardId>
<href>http://{server-url}/Dashboard/#desktop/EmsF8TH6KrAx3UbYZxqqm5/1</href>
         <name>Dashboard 2</name>
         <resourceId>E123</resourceId>
      </dashboard>
   </items>
   <fullSize>2</fullSize>
   <limit>100</limit>
   <offset>0</offset>
</dashboardsResult>
```

7.2 List dashboard templates

Lists dashboard templates from enterprises. Shows also shared dashboard templates.
URL: /templates/dashboard
HTTP method: GET
Authentication Required: Yes

Parameter	Description	Example
format	Format (json or xml) default is json	/templates/dashboard?format=xml
limit (integer)	Number of results to return. Maximum of 100, defaults to 100 when none is provided	/templates/dashboard?limit=5
offset (integer)	Number of results to skip from the beginning.	/templates/dashboard?offset=3 Paging example: /templates/dashboard?offset=0&limit=100 /templates/dashboard?offset=100&limit=100
enterpriseIds	The ids of the enterprises to fetch the templates from (with or without the "E" prefix).	/templates/dashboard?enterpriseIds=123,E234

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error handling chapter.

Response to list dashboard templates

Field	Description
offset	Number of results skipped from the beginning
limit	Number of results per page
fullSize	The total number of dashboard templates found for the request.
items	List of dashboard template objects. Each of the objects contain • dashboardTemplateId • name • enterpriseId • activeVersion
dashboardTemplateId	The id of the dashboard template.
name	The name of the dashboard template.
enterpriseId	The resource id of the enterprise the dashboard template belongs to. (with shared templates, this shows the real source enterpriseId unless the user does not have access to that enterprise. In that case, enterpriseId is written as "shared")
activeVersion	The current active version of the dashboard template.

Example

GET Request
[**https://\{server-url}/templates/dashboard?enterpriseIds=E123** **REQUEST PAYLOAD: NONE**
JSON Response

```
{
  "fullSize": 2,
  "limit": 100,
  "offset": 0,
  "items": [
    {
      "dashboardTemplateId": 123,
      "name": "Template 1",
      "enterpriseId": "E123",
      "activeVersion": "2"
    },
    {
      "dashboardTemplateId": 234,
      "name": "Template 2",
      "enterpriseId": "E123",
      "activeVersion": "V1"
    }
  ]
}
```

XML Response

```
<dashboardTemplatesResult>
  <items>
    <dashboardTemplate>
      <dashboardTemplateId>123</dashboardTemplateId>
      <name>Template 1</name>
      <enterpriseId>E123</enterpriseId>
      <activeVersion>2</activeVersion>
    </dashboardTemplate>
    <dashboardTemplate>
      <dashboardTemplateId>234</dashboardTemplateId>
      <name>Template 2</name>
      <enterpriseId>E123</enterpriseId>
      <activeVersion>V1</activeVersion>
    </dashboardTemplate>
  </items>
  <fullSize>2</fullSize>
  <limit>10</limit>
  <offset>0</offset>
</dashboardTemplatesResult>
```

7.3 Create dashboard

Creates dashboard based on a template to a resource.
URL: /resources/dashboards

HTTP method: POST

Authentication Required: Yes

Parameter	Description	Example
Object	A Json object with the following fields: • name • resourceId • templateId • createMode (optional) • viewMode (optional)	`{ "name": "Dashboard 1", "resourceId": "X123", "templateId": 234, "createMode": 1, "viewMode": 1 }`
resourceId	The target resource id.	X123
name	Name for the new dashboard	Dashboard 1
templateId	The dashboard template id to be used to create the dashboard	234
createMode (optional)	This determines the logic for verifying if the dashboard needs to be created to the target resource. Select either • **0** to create always (default) • **1** to create only if dashboard from selected template does not already exist in the target resource	0 (as integer)
viewMode (optional)	This determines who can see the created dashboard. Select either • **0** for users with manager permissions for the target resource or if the dashboard is shared to them (default) • **1** for possibility to view the dashboard to all users who can see the resource	1 (as integer)

When any of the required fields are not present or the parameter constraints are violated, an HTTP status code 400 is returned along with an error object in the response payload. For more details see the Error handling chapter.

Response to create dashboard

Field	Description
dashboardId	Dashboard id of the created dashboard.
name	Name of the created dashboard.
resourceId	The resource id of the resource the dashboard belongs to.
templateId	The dashboard template id of the created dashboard.
templateName	The dashboard template name of the created dashboard.

Field	Description
href	Link to the dashboard in the IoT-Ticket accessible by web browser.
created	Boolean value if a new dashboard was registered (only present when createMode 1 is used).

Example

POST Request
[**https://\{server-url}/resources/dashboards** **REQUEST PAYLOAD is shown in the example above**
JSON Response

```
{
   "dashboardId": "6z83eeNLnf2Ur4Cda5tRY8",
   "dashboardTemplateId": 234,
   "dashboardTemplateName": "Template 1",
   "href": "https://{server-url}/Dashboard/#desktop/6z83eeNLnf2Ur4Cda5tRY8/1",
   "name": "Dashboard 1",
   "resourceId": "X123"
}
```

8 Error handling

Errors to the API are appearing as a result of invalid credentials, unauthorized access to the target resource, data format issues and sometimes internal server problems. Nonetheless, the API always returns one of the following HTTP Status Codes:

- 200 – OK
- 201 – Created
- 400 – Bad Request
- 401 – Unauthorized
- 403 – Forbidden
- 500 – Internal Server error

In addition to checking the HTTP Status, developers should also view the response body entity that will describe the error further, if any of the statuses above is received, except the OK status.

Field	Description
description (String:500)	This field provides a general description of the error.
code (int)	See the Error codes table below for the codes and their meaning.
moreInfo (String:255)	This field points to the documentation URL where a more detailed description about the error code can be found.
apiver (int)	The API version number.

Example

Response (HTTP STATUS 400)

```
{
  "description":"Request cannot be processed because you have exceeded the limit of 30 Megabytes.
Visit the WRM website to increase your allotted storage size",
  "code":8002,
  "moreInfo":"https://{server-url}/errorcodes",
  "apiver":1
}
```

Error codes and their meanings

Code	Meaning
8000	Internal server error.
8001	Permission on requested resource is not sufficient.
8002	Quota violated.

Code	Meaning
8003	Bad Parameters provided.
8004	Write failed.

9 Quota Management

When the client quota is exceeded, the API will return an HTTP response 403 Forbidden, even though the request is valid. A message object is also returned that includes specific details on which aspect of the quota has been violated. Note that in calculating storage size, 1 Megabyte of data is regarded as 1048576 bytes and only the size of the values are calculated in the quota.

9.1 Get overall quota

URL: /quota/all
HTTP method: GET
Authentication Required: Yes
The call returns an overview of the client's resource usage, i.e. how much resources have been used and the maximum amounts allowed.

Parameter	Description	Example
callback	The JavaScript function to be executed when the response is received (JSONP).	/quota/all?callback=foo returns: foo(response data);
format	The format: json or xml; the default format is json.	/quota/all?format=xml

Response to get the overall quota

Field	Description
totalDevices (integer)	Total number of devices the client owns. ⓘ This is not the same as the total number of devices the client has access to.
maxNumberOfDevices (integer)	The maximum number of devices the client can create.
maxDataNodePerDevice (integer)	The maximum of number of devices allowed for a client per datanode.
usedStorageSize (long)	The total size in bytes that the client has written to the server.
maxStorageSize (long)	The maximum size in bytes that the client has a right to write to the server.

Example

Request
[**https://\{server-url}/quota/all** **REQUEST PAYLOAD:NONE**
JSON Response

```
{
    "totalDevices": 3,
    "maxNumberOfDevices": 5,
    "maxDataNodePerDevice": 10,
    "usedStorageSize": 1048576,
    "maxStorageSize": 52428800
}
```

XML Response

```
<quota>
    <totalDevices>3</totalDevices>
    <maxNumberOfDevices>5</maxNumberOfDevices>
    <maxDataNodePerDevice>10</maxDevicePerDataNode>
    <usedStorageSize>1048576</usedStorageSize>
    <maxStorageSize>52428800</maxStorageSize>
</quota>
```

9.2 Get device specific quota

URL: /quota/{deviceId}

HTTP method: GET

Authentication Required: Yes.

Parameter	Description	Example
callback	The JavaScript function to be executed when the response is received (JSONP).	/quota/all?callback=foo returns: foo(response data);
format	The format: json or xml; the default format is json.	/quota/all?format=xml

Response to get a device specific quota

Field	Description
totalRequestToday (integer)	The total number of requests made through the API to the device. Any serviced URL that includes the device ID is added to this count.
maxReadRequestPerDay (integer)	The maximum number of read requests allowed to the client for this device.
numberOfDataNodes (integer)	The number of datanodes created for the specific device.
storageSize (long)	The total size in bytes that the client has written to the server for the specific device.
deviceId (String:32)	The device ID.

Example

Request

[https://\{server-url}/quota/{deviceId}
REQUEST PAYLOAD:NONE

JSON Response

```
{
   "deviceId":"258d5f5cf04446199f7b754c25dae257"
   "totalRequestToday": 5000,
   "maxReadRequestPerDay": 100000,
   "numberOfDataNodes": 5,
   "storageSize": 3072
}
```

XML Response

```
<quota>
   <totalDevices>3</totalDevices>
   <maxDeviceAllowed>5</maxDeviceAllowed>
   <maxDataNodePerDevice>10</maxDevicePerDataNode>
   <usedStorageSize>1048576</usedStorageSize>
   <maxStorageSize>52428800</maxStorageSize>
</quota>
```

Yliopistonranta 5

65200 Vaasa

+358 10 277
5000

support@iot-ticket.com

www.IoT-TICKET.com

Map data ©2016 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional
Terms of Use