# SMART SYSTEMS REST API QUERIES

## System Queries

### get Time

Usage: <http://IP:Port/api/getTime>

Description: Returns the local time on the server

## Asset Queries

### Get Asset Hierarchy

Usage: <http://IP:Port/api/getAssetHierarchy>

Description: Returns the smart systems asset hierarchy

### Get Asset Types

Usage: <http://IP:Port/api/getAssetTypes>

Description: Returns the smart systems asset types

### Get Assets

Usage: <http://IP:Port/api/getAssets>

Description: Returns the smart systems assets

### Create Asset

Usage: [http://IP:Port/api/createAsset?type={Asset Type}&name={Asset Name](http://IP:Port/api/createAsset?type=%7bAsset%20Type%7d&name=%7bAsset%20Name)}

Example: <http://IP:Port/api/createAsset?type=Pump&name=PumpABC>

Description: Creates a new asset based on Pump Type

### Rename Asset

Usage: [http://IP:Port/api/renameAsset?name={Current Asset Name}&newname={New Asset Name](http://IP:Port/api/renameAsset?name=%7bCurrent%20Asset%20Name%7d&newname=%7bNew%20Asset%20Name)}

Example: [http://IP:Port/api/ renameAsset?type=PumpABC&name=PumpXYZ](http://IP:Port/api/createAsset?type=PumpABC&name=PumpXYZ)

Description: Renames the Asset

### DELETE Asset

Usage: [http://IP:Port/api/deleteAsset?name={Asset Name](http://IP:Port/api/deleteAsset?name=%7bAsset%20Name)}

Example: <http://IP:Port/api/deleteAsset?name=PumpXYZ>

Description: Deletes Asset

## Nameplate Queries

### Get Nameplate

Usage: [http://IP:Port/api/getNameplate?name={Asset Name](http://IP:Port/api/getNameplate?name=%7bAsset%20Name)}

Example: <http://IP:Port/api/getNameplate?name=fan>

Description: Returns nameplate information of the selected equipment. Web client can only change the Value and send it back to Smart Systems using setNameplate query which is explained in the next section. The value changed by user through web client has to be between Min and Max values. If the Min or Max value is NaN, there is no need to check the corresponding limit. If both Min and Max are NaN, there is no need to check any of them. Set Nameplate already checks the limits. However, it is strongly recommended to check the values before submitting them to Smart System.

[

    {

        "Value": 5000.0,

        "Unit": "Hz",

        "Min": 1000.0,

        "Max": 32000.0,

        "Method": "",

        "Parameters": [],

        "Type": 0,

        "Calculated": **false**,

        "Enabled": **true**,

        "Assembly": "Core",

        "Name": "SR",

        "Title": "Sampling Rate"

    },

    {

        "Value": 10.0,

        "Unit": "s",

        "Min": 1.0,

        "Max": 20.0,

        "Method": "",

        "Parameters": [],

        "Type": 0,

        "Calculated": **false**,

        "Enabled": **true**,

        "Assembly": "Core",

        "Name": "Duration",

        "Title": "Duration"

    },

    {…………………………………..}]

### Set nameplate

Usage: [http://IP:Port/api/setNameplate?name={Asset Name](http://IP:Port/api/setNameplate?name=%7bAsset%20Name)}

Description: Overwrites the nameplate values changed by the user. User is not allowed to change any other parameters than the value by checking the Min and Max values. User can update a single nameplate parameter by sending a jsonobject in the body of the query or alternatively send an array of nameplate values as shon in the example below.

Example: <http://IP:Port/api/setNameplate?name=fan>

* Open Postman Click "New Request"
* Set Method & URL POST http://localhost:5000/api/ setNameplate?name=fan
* Content-Type: application/json
* Add JSON Body

[

    {

        "**Value": 8000.0,**

        "Unit": "Hz",

        "Min": 1000.0,

        "Max": 32000.0,

        "Method": "",

        "Parameters": [],

        "Type": 0,

        "Calculated": **false**,

        "Enabled": **true**,

        "Assembly": "Core",

        "Name": "SR",

        "Title": "Sampling Rate"

    },

    {

        "Value": 5.0,

        "Unit": "s",

        "Min": 1.0,

        "Max": 20.0,

        "Method": "",

        "Parameters": [],

        "Type": 0,

        "Calculated": **false**,

        "Enabled": **true**,

        "Assembly": "Core",

        "Name": "Duration",

        "Title": "Duration"

    },

    {…………………………………..}]

* Click "Send"

## File Queries

### Get json File

Usage: [http://IP:Port/api/getJsonFile?name={File Name](http://IP:Port/api/getJsonFile?name=%7bFile%20Name)}

Example: <http://IP:Port/api/getJsonFile?name=settings>

Description: Returns the requested Json file. There are two Json files can be requested. These are ‘settings’ and ‘project’.

### Set json File

Usage: [http://IP:Port/api/setJsonFile?name={File Name](http://IP:Port/api/setJsonFile?name=%7bFile%20Name)}

Example: <http://IP:Port/api/getJsonFile?name=settings>

* Open Postman Click "New Request"
* Set Method & URL POST http://localhost:5000/api/setJsonFile?name=settings
* Content-Type: application/json
* Add JSON Body

{

"DBUrl": "http://localhost:8086",

"DBToken": "uR00uDuxtmFqzlvCU00-0f1G0NYY5dl5Ao0dkt0K\_WRqmHz3kOe7vg8r9QCfC8rqeg4V5ATkgd7eRoFlZOSiHA==",

"RetentionPeriod": 63072000,

"Company": 1,

"AlertQueueSize": 6,

"RemoveOutliers": true,

"OutlierSigma": 16.0,

"Sigmas": [-100000.0, -14.0, -10.0, -6.0, 6.0, 10.0, 14.0, 100000.0],

"CountThresholds": [2.0, 4.0, 8.0],

"BoundaryScale": [1.0, 0.05, 1.0, 1.0],

"Debug": true,

"DueDays": 31,

"FaultLimits": [-0.5, 5.0, 10.0, 20.0, 100000.0],

"DateNavigatorShiftDays": 61,

"TrendRecordCount": 1000,

"logLevel": 0,

"logDaysToKeep": 7,

"backupDaysToKeep": 365,

"FileProcessType": 1,

"DataPath": "C:\\Users\\Public\\Documents\\SmartSystems\\data\\"

}

* Click "Send"

Description: Overwrites the requested Json file. Settings file can be overwritten but project file is read only. Therefore, project file will not be overwritten with this command.

## Diagnostic Queries

### Get Diagnostic

Usage: [http://IP:Port/api/getDiagnostic?name={Asset Name](http://IP:Port/api/getDiagnostic?name=%7bAsset%20Name)}

Example: [http://IP:Port/api/ getDiagnostic?name=fan](http://IP:Port/api/%20getDiagnostic?name=fan)

A screenshot of a computer program

AI-generated content may be incorrect.

Description: This query returns the diagnostic information in two separate sections. These are BarGraph and Tree List.

* ParentID and ID are used to create hierarchical structure.
* Names and Titles can be used to display the parameters in Bar graph and tree list. Name is the short form of the title.
* Status value indicates the Status of the parameters.
* Value is the value of the parameter. If the Value is NaN, that means this parameter cannot have a value to display. For instance, none of the BarGraph parameters will have a value. All of them will be NaN because they are displayed based on Status. Diagnostic status can be one of the following values.

Status: NoData=0, OK = 1, Warning=2, Inspect=3, Repair=4

* ListTree provides more information and it includes sub parameters. The top level in tree list is the diagnostic parameter (such as Load, Ground Fault, Unbalance, etc.). Under each diagnostic parameter, the parameters which are used to calculate the diagnostic parameter will be listed. Parameters can have sub parameters. For instance, rmsi is a parameter which feeds Load diagnostic parameter and rmsi has three sub parameters rmsiA, rmsB, and rmsC. Rmsi parameters status and value is calculated from the max of status and value of its sub parameters.

A screenshot of a computer

AI-generated content may be incorrect.

### Get Power quality

Usage: [http://IP:Port/api/getPQ?name={Asset Name](http://IP:Port/api/getPQ?name=%7bAsset%20Name)}

Example: [http://IP:Port/api/ getPQ?name=fan](http://IP:Port/api/%20getPQ?name=fan)

A screenshot of a computer

AI-generated content may be incorrect.

Description: This query returns the power quality information. It is structured in exactly the same way with Diagnostic. The only difference is the Status of the power quality parameters use the following status values.

Status: NoData = 0, OK = 1, Low=2, Medium=3, High=4

A screenshot of a computer

AI-generated content may be incorrect.

## User Management Queries

### Login

Usage: [http://IP:Port/api/login?username={UserName}&password={Password}](http://IP:Port/api/login?username=%7bUserName%7d&password=%7bPassword%7d)

Example: <http://IP:Port/api/login?username=admin&password=admin>

### Logoff

Usage: <http://IP:Port/api/logoff>

Example: <http://IP:Port/api/logoff>

### Get Users

Usage: <http://IP:Port/api/getUsers>

Example: <http://IP:Port/api/getUsers>

### Create User

Usage: [http://IP:Port/api/createUser?username={UserName}&password={Password}&level={Level}](http://IP:Port/api/createUser?username=%7bUserName%7d&password=%7bPassword%7d&level=%7bLevel%7d)

Example: [http://IP:Port/api/createUser?username=user1&password=user1&level=”0](http://IP:Port/api/createUser?username=user1&password=user1&level=)”

User Levels: Admin:0, Operator: 1, User: 2

### Change User Level

Usage: [http://IP:Port/api/changeUserLevel?username={UserName}&level={Level}](http://IP:Port/api/changeUserLevel?username=%7bUserName%7d&level=%7bLevel%7d)

Example: [http://IP:Port/api/changeUserLevel?username=user1&level=”1”](http://IP:Port/api/changeUserLevel?username=user1&level=)

### Delete User

Usage: [http://IP:Port/api/deleteUser?username={UserName}](http://IP:Port/api/deleteUser?username=%7bUserName%7d)

Example: <http://IP:Port/api/login?username=user1>

### Change Password

Usage: [http://IP:Port/api/changePassword?username={UserName}&oldpassword={OldPassword}&password={Password}](http://IP:Port/api/changePassword?username=%7bUserName%7d&oldpassword=%7bOldPassword%7d&password=%7bPassword%7d)

Example: <http://IP:Port/api/changePassword?username=user1&oldpassword=user1&password=user2>

## Trending Queries

### Get Trend Parameters Hiearchy

Usage: <http://IP:Port/api/getTrendHierarchy>

Example: [http://IP:Port/api/getTrendHierarchy](http://IP:Port/api/getTrendHierarchy%20)

This query returns the parameters hierarchy as shown in the json data below. Client application should use this json data to create the parameters hierarchy and let the use select PARAMETERs=1 or EXTPARAMETERs=11 and send these selected parameters to GetTrendData command to get the trend data for the corresponding parameters. TREENODETYPE is the enumerator for NodeType.

*public enum TREENODETYPE { NONE, MASTER, RECYCLEBIN, ROOT, NODE, MACHINE, ASSET, ASSEMBLY, DIAGNOSTIC, POWERQUALITY, PARAMETER, EXTPARAMETER, FAULT, EVENT, MAINTENANCE, DATABASE, TEMPLATE }*

[{

"ParentID": -1,

"ID": 1,

"Name": "Plant",

"Title": "Plant",

"Status": 0,

"Value": "NaN",

"NodeType": 3,

"Path": "",

"Acknowledged": false

}, {

"ParentID": 1,

"ID": 2,

"Name": "Fan",

"Title": "Fan",

"Status": 0,

"Value": "NaN",

"NodeType": 6,

"Path": "Fan",

"Acknowledged": false

}, {

"ParentID": 2,

"ID": 3,

"Name": "RF",

"Title": "Rotation Frequency ",

"Status": 0,

"Value": 0.0,

"NodeType": 11,

"Path": "Fan",

"Acknowledged": false

}, {

"ParentID": 2,

"ID": 4,

"Name": "Motor",

"Title": "Motor",

"Status": 0,

"Value": "NaN",

"NodeType": 7,

"Path": "Fan | Motor",

"Acknowledged": false

}, {

"ParentID": 2,

"ID": 5,

"Name": "VFRatio",

"Title": "VF Ratio ",

"Status": 0,

"Value": 0.0,

"NodeType": 11,

"Path": "Fan | Motor",

"Acknowledged": false

},

{

"ParentID": 2,

"ID": 10,

"Name": "Slip",

"Title": "Slip ",

"Status": 0,

"Value": 0.0,

"NodeType": 11,

"Path": "Fan | Motor",

"Acknowledged": false

}, {

"ParentID": 2,

"ID": 11,

"Name": "RF",

"Title": "Rotation Frequency ",

"Status": 0,

"Value": 0.0,

"NodeType": 11,

"Path": "Fan | Motor",

"Acknowledged": false

}, {

"ParentID": 2,

"ID": 13,

"Name": "harmonicsi",

"Title": "Current Harmonic",

"Status": 0,

"Value": 0.0,

"NodeType": 10,

"Path": "Fan | Motor | PowerSupply",

"Acknowledged": false

}, {

"ParentID": 2,

"ID": 22,

"Name": "rmsi",

"Title": "Current RMS",

"Status": 0,

"Value": 0.0,

"NodeType": 10,

"Path": "Fan | Motor | PowerSupply",

"Acknowledged": false

}, {

"ParentID": 2,

"ID": 26,

"Name": "PercentLoadAmp",

"Title": "Percent Load Amp ",

"Status": 0,

"Value": 0.0,

"NodeType": 11,

"Path": "Fan | Motor | PowerSupply",

"Acknowledged": false

}]

### Get Trend Data (body-Raw-json : post)

Usage: <http://IP:Port/api/getTrendData>

Example: <http://IP:Port/api/getTrendData>

Create the json data below to get the trend data from Smart Systems. Client application should set a start date and end date for the query and also parameter ids which are selected by the user.

{

"ParametersIds": [13, 28],

"StartDate": "2024-11-22T21:28:41.9779368-05:00",

"EndDate": "2025-03-24T21:28:41.9779445-04:00"

}

C# Class for Trend Request:

public class TrendRequest

{

public TrendRequest() { }

public List<int> ParametersIds { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

}

**Response:**

Response is return as **CombinedTrendData.**

If Threshold is null or empty, that means there is no threshold data. Therefore, threshold should not be plotted in this case.

Trend Data includes data to plot for each parameter requested.

TreeItem in Trend data includes the information about the parameter like parameter name, title, unit etc.

* By using sample data from the API documentation, if you send NodeType=11 as a parameter, a threshold will be returned.

**public class CombinedTrendData**

{

public CombinedTrendData() { Parameters = new List<TrendData>(); }

public List<TrendData> Parameters { get; set; }

public List<TrendThresholds> Thresholds { get; set; } = null;

}

**public class TrendData**

{

public TreeItem UID { get; set; }

public List<DataPointTime> Data { get; set; }

}

**public class DataPointTime**

{

public DataPointTime() { }

public DateTime XAxis { get; set; }

public double? YAxis { get; set; }

}

Threshold:

YAxis: Thr1,ThrM,ThrN

XAxis:Date1,DateM, DateN

You need to use the same threshold (Thr1) between Date1 to DateM-1

You need to use the same threshold (ThrM) between DateM to DateN-1

You need to use the same threshold (ThrN) between DateN to DateN

public static readonly string[] ThresholdRange = new string[] {"Out Of Range", "Repair", "Inspect", "Warning", "Warning", "Inspect", "Repair", "Out Of Range"};

## Service Operations

### Service Operations

Usage: [http://IP:Port/api/serviceOp&cmd={Command}&name={ServiceName}](http://IP:Port/api/serviceOp&cmd=%7bCommand%7d&name=%7bServiceName%7d)

Example: <http://IP:Port/api/serviceOp&cmd=start&name=db>

This query sends the command to Database Service (db) or Smart System Application Service (app)

**Commands:** start, stop, restart, status

**Service Name:** db, app

## Downloads

### Get Folder

Usage: [http://IP:Port/api/getFolder&name={FolderName}](http://IP:Port/api/getFolder&name=%7bFolderName%7d)

Example: <http://IP:Port/api/getFolder&name=dbbackup>

**Folder Name:** all, log, project, backup, dbbackup

This query compresses the requested folder and returns as a zip file.

**Folder Names:**

**Dbbackup:** backups influxdb, compress the backup file to a single zip file and sends back to client

**Log:** compress the log folder including all the log files to a single zip file and sends back to client

**Project:** compress the project folder to a single zip file and sends back to client

**Backup:** compress the project backup folder to a single zip file and sends back to client

**All:** compress database, log, project, and backup folders to a single zip file and sends back to client

## Faults & Events

### Get FauLts

Usage: [http://IP:Port/api/getFaults? name={Asset Name](http://IP:Port/api/getFaults?%20name=%7bAsset%20Name)}

Example: [http://IP:Port/api/getFaults? name=fan](http://IP:Port/api/getFaults?%20name=fan)

Description: Returns the faults in Bar Graph and Tree List format similar to GetPQ and GetDiagnostics

### Get Events

Usage: [http://IP:Port/api/getEvents? name={Asset Name](http://IP:Port/api/getEvents?%20name=%7bAsset%20Name)}

Example: [http://IP:Port/api/getEvents? name=fan](http://IP:Port/api/getEvents?%20name=fan)

Description: Returns the faults in Bar Graph and Tree List format similar to GetPQ and GetDiagnostics