

RFC-11 : REST API specification

Author: Guillaume, Gerald

Version: **V1**

Status: **IN-PROGRESS**

- 1 About
- 2 Global architecture
 - 2.1 Operations summary
 - 2.1.1 Available operations
 - 2.1.2 Statuses of the operations
 - 2.1.3 Terminology
 - 2.1.4 Version
 - 2.1.5 URL parameters
 - 2.1.6 Formats
 - 2.1.7 URI format Namespace/Application
 - 2.1.7.1 Available categories
 - 2.2 Security
 - 2.2.1 Examples:
 - 2.2.1.1 Obtain a token
 - 2.2.1.2 Do an authorized request
 - 2.2.1.3 Revoke the token
 - 2.3 Header
 - 2.4 HTTP status codes
 - 2.5 Error handling
 - 2.5.1 Error message template
 - 2.5.2 Error codes and error message template
 - 2.6 Response/Request Documents
- 3 Operation details
 - 3.1 Asset (v1)
 - 3.1.1 Asset types and subtypes possible combinations
 - 3.1.2 Extended attributes consideration
 - 3.1.3 export csv(Read)
 - 3.1.4 import csv(Create)
 - 3.1.5 Create one asset element
 - 3.1.6 Delete one asset element
 - 3.1.7 Update one asset element
 - 3.1.8 assets (Read)
 - 3.1.9 devices, datacenters, rooms, racks, rows (Read)
 - 3.1.10 datacenter (Read)
 - 3.1.11 room(Read)
 - 3.1.12 row(Read)
 - 3.1.13 groups (Read)
 - 3.1.14 group(Read), (Create, Update) Not Yet Implemented
 - 3.1.15 rack(Read)
 - 3.1.16 device(Read)
 - 3.2 Topology (v1)
 - 3.2.1 location (Read)
 - 3.2.2 power (Read)
 - 3.2.3 input_power_chain (Read)
 - 3.3 Metric(v1)
 - 3.3.1 current(Read)
 - 3.3.2 Outlet properties
 - 3.3.3 computed
 - 3.3.3.1 rack_total(Read)
 - 3.3.3.2 datacenter_indicators (Read)
 - 3.3.3.3 average (Read) - Work in Progress
 - 3.3.3.4 uptime(Read)
 - 3.4 Alert (v1)
 - 3.4.1 activelist (Read)
 - 3.4.2 rules (Read, Create, Update)
 - 3.4.3 acknowledge (Update)
 - 3.5 Admin (v1)
 - 3.5.1 time (Read, Update)
 - 3.5.2 config (Read, Update)
 - 3.5.2.1 Read
 - 3.5.2.2 Update
 - 3.5.3 systemctl
 - 3.5.3.1 list (Read)

- 3.5.3.2 [status](#) (Read)
 - 3.5.3.3 [start](#) (Update)
 - 3.5.3.4 [stop](#) (Update)
 - 3.5.3.5 [restart](#) (Update)
 - 3.5.3.6 [enable](#) (Update)
 - 3.5.3.7 [disable](#) (Update)
 - 3.5.3.8 [Managing networks via /admin/systemctl](#)
- 3.5.4 [netcfgs](#) (Read)
- 3.5.5 [netcfg](#) (Read, Update)
- 3.5.6 [ifaces](#) (Read)
- 3.5.7 [iface](#) (Read)
- 3.5.8 [getlog](#) (Read)
- 3.5.9 [license](#) (Read, Update)
- 3.5.10 [passwd](#) (Update)
- 3.5.11 [sysinfo](#) (Read)
- 3.5.12 [email/test](#) (Create)
- 3.5.13 [email/feedback](#) (Create)
- 3.6 [Special email headers](#)
- 3.7 [email/vote](#)
- 3.8 [Special email headers](#)
- 3.9 [OAuth2 \(v1\)](#)
 - 3.9.1 [token](#)(Create)
 - 3.9.2 [revoke](#) (Update)

About

This RFC describes REST API calls supported by the system. This API allows the user to retrieve any collected information about datacenters monitored by RC.

Note: WS - web service

Global architecture

The following web services are designed according RESTful architecture.

RESTful APIs are stateless and agnostic, several kinds of clients are supported.

Operations summary

Available operations

RESTful architecture supports the following set of operation: create, read, update, delete.

These methods corresponds to the HTTP methods as described in the table bellow:

Operation	HTTP method	Description
Create	POST	Creates a new resource
Read	GET	Retrieves an information about requested resource
Update	PUT	Updates an information about existing resource
Delete	DELETE	Deletes a specified resource

Statuses of the operations

Status	Description
Actual	The functionality of the url is fully implemented and supported. It can be used in current development

Deprecated	The functionality of the url is fully implemented, but in future this will be removed. It is not recommended to use in current development.
Not implemented	The functionality is still not implemented

Terminology

<param-id> : "param-id" is a parameter, the whole sequence is replaced by the value of a parameter.

[text] : this sequence means optional sequence.

a|b : the pipe means a OR b

Version

The version of the API is included into URL.

As an example /api/v1/category/method means the version "v1" of the category.

URL parameters

All URL parameters should be in lower case.

Good:

- /api/v1/topology/location?from=none
- /api/v1/topology/location?from=ThisIsID1222

Wrong:

- /api/v1/topology/location?FROM=none
- /api/v1/topology/location?frOm=none

Formats

- Format of the response document is specified for each WS.
- Date format
 - Several formats maybe considered:
 - ISO 8601 (YYYY-MM-DDThh:mm:ssZ).
 - Extended ISO 8601 YYYY-MM-DDThh:mm:ss.sssZ
 - ISO 8601 version without hyphens and colons, such as: YYYYMMDDThhmmssZ
 - Epoch time such as 1331162374
 - In this early version, the preferred one is ISO 8601
 - For now each WS defines it in its own way

URI format Namespace/Application

The URI uses the format [http\(s\)://<rc_address>/api/vx/<category><extra_param>](#) where :

UPI part	Description
http(s)://<rc_address>	Rack Controller address
/api	REST API entry point
/vx	category version, where x is a number
/<category>	application name or namespace
/<extra_param>	extra parameters depends on category

Available categories

Category	Description
asset	retrieve and modify assets
metric	retrieve metrics, indicators, detailed historical measurements
topology	retrieve topologies of the datacenter
alert	retrieve, acknowledge and clear alerts
admin	RC administration stuff
oauth2	token management
ui	for ui client usage

Security

Two levels of security are available :

- public access
- restricted access

A public access does not require any login. A public access may be done under HTTP or HTTPS

A restricted access requires a valid credential, a correct role and a secure connection. Only HTTPS is available (in production builds; the `DEBUG` builds allow non-HTTPS authorized access but report it to the log file).

A valid credential should be a valid login/password or a valid token. Credentials should be always transmitted under a secure connection.

For certain services, even reading requests (not only modifications) to sensitive data are only satisfied with a sufficiently high access level.

On programmatic level, negative access level values are also defined to signify an authorization error. This is intended to facilitate audit logging and other error handling; as far as service access is concerned, this is processed as "public" access with `access_auth_level<=0`, and if access to sensitive services is requested where `access_auth_level>0` is required, this is a good reason to redirect the client to the login service URL instead of serving the request.

In this document, different roles are used to specify the minimum role level access to get access an interface :

access_auth_level number	Effective access	Description
-2	public = badtoken	a token or header for login was presented by client, and the <code>access_token</code> string was not empty, but it was an invalid token (did not pass <code>verify_token()</code> successfully) – perhaps a left-over of an expired session, etc.
-1	public = badtoken	a token or header for login was presented by client, but the <code>access_token</code> resolved to an empty string NOTE: This is currently implemented only for <code>access_tokens</code> passed via header mechanism; specified but empty GET/POST parameters are currently ignored and result in <code>access-auth_level==0</code> .
0	public = guest	guest user with no modification permission, it corresponds to the public access
1	restricted = user	user with restricted modification permission
2	restricted = poweruser	user with high level of permission
3	restricted = admin	administrator with full rights

Level 1 and level 3 are not yet supported (that is, any successful authentication is currently hardcoded to be a "level=2" authorization).

For development/debugging/testing purposes, the `DEBUG` builds of the \$BIOS core REST API server expose the authorization-related variables in the response header, so a developer can validate the server's decision about the provided authentication, as well as hint at

which code-paths (source files) were involved in construction of the particular HTTP response. Such report headers MAY include:

Header	Example value	Description	Set by source file...
X-Auth-Verify	"0"	Numeric value of <code>access_auth_level</code> set after the interpretation of <code>access_token</code> string from the request	auth-verify.ecpp
X-Auth-Token-Presented	"" "asd/123+12s/a"	String value of <code>access_auth_level</code> as used for interpretation of the request (after un-escaping spaces back into pluses, for example)	auth-verify.ecpp
X-Auth-Token-PassedBy	"none" "auth_header_notBearer" "auth_header_empty" "auth_header_value" "query_params_value"	The final one of several possible sources that provided the <code>access_token</code> string value (if several conflicting values were passed, which one did we ultimately trust?)	auth-verify.ecpp
X-Auth-Require	2	Numeric value of <code>access_auth_level</code> passed into the handler which requires a valid session or redirects to login URL (for POST/PUT/DELETE requests)	auth-require.ecpp
X-Auth-SysInfo	-2	Numeric value of <code>access_auth_level</code> passed into the handler for <code>.../admin/sysinfo</code> which produces different outputs based on auth level (this service in <code>DEBUG</code> builds may also provide some of the information above as part of its output markup, though that optional feature should not be relied upon)	sysinfo.ecpp

This functionality is not available in final production builds.

OAuth 2.0 token credential with SSL is the preferred token authentication mechanism supported by \$BIOS.

Examples:

The workflow to get and use a token is simple. See bellow.

Obtain a token

GET or POST to `/api/v1/oauth2/token` service:

```
curl -v --progress-bar /api/v1/oauth2/token?username=<user>&password=<pwd>&grant_type=password
```

Do an authorized request

In the HTTP header, add `Authorization: Bearer $token` string:

```
HEADER: curl -v -H "Authorization: Bearer $TOKEN" "${REQUESTURL}?${REQUESTPARAMS}"
```

...or do a GET/POST request including the parameter `access_token=$TOKEN` explicitly:

```
GET:      curl -v "${REQUESTURL}?${REQUESTPARAMS}&access_token=$TOKEN"
POST:     curl -v -d "${REQUESTPARAMS}&access_token=$TOKEN"
"${REQUESTURL}"
```

Revoke the token

```
curl -v --insecure -d 'token=$TOKEN' -H "Authorization: Bearer $TOKEN" -X POST
'/api/v1/oauth2/revoke'
```

Header

To make a request with JSON, the appropriate HTTP header is :

Content-Type: application/json
Accept: application/json

The default format is JSON.

HTTP status codes

code	Description
Success codes	
200	Successful operation
Error codes	
400	Bad input parameter. Error message should indicate which one and why
401	Unauthorized. You need valid credentials
403	Forbidden. Credentials are Ok, but you're not allowed
404	Resource not found. Typically, resource id is incorrect.
405	Method not allowed
409	Indicates that the request could not be processed because of conflict in the request
413	The server is refusing to process a request because the request entity is larger than the server is willing or able to process.
418	Undefined error (TEAPOT). You should never see this error to be returned. If so, please report it to the team. So we can fix the problem.
500	Internal Error
501	Not Implemented

In our project we use only one return code that indicates a success : HTTP 200.

Error handling

Inspired by <https://dev.twitter.com/overview/api/response-codes>

Error message template

```
{
  "errors": [
    {
      "message" : "<error_message>",
      "code" : <error_code>
    }
  ]
}
```

Error codes and error message template

To be consistent with [Messages Library](#)

As %s some additional information can be passed

Code	Message(English)	Corresponding HTTP status code	Additional description/comments
42	Internal Server Error. %s	500 Internal server error	
43	You are not authorized. Please use '/oauth2/token?username=<user_name>&password=<password>&grant_type=password' GET request to authorize.	401 Unauthorised	
44	Element '%s' not found.	404 Not found	
45	Http method '%s' not allowed.	405 Method not allowed	
46	Parameter '%s' is required.	400 Bad request	
47	Parameter '%s' has bad value. Received %s. Expected %s.	400 Bad request	
48	Request document has invalid syntax. %s	400 Bad request	
50	Element '%s' cannot be processed because of conflict. %s	409 Conflict	
51	%s is forbidden. %s"	403 Forbidden	
52	Request cannot be processed because of conflict in parameters. %s	400 Bad request	
53	Content size is too big, maximum size is %s	413 Request entity is too large	
54	%s does not exist.	404 Not found	

Response/Request Documents

For each WS it is specified separately, what format is used:

- JSON
- csv

Most usable format is JSON.

Current limitations:

- Request document is read from the beginning. Correct JSON document is read. If there is left some trash, it would be ignored. If request document is

```
{ "aa" : "1" }, slkfjsdlfkj
```

Then it would be interpreted as

```
{ "aa" : "1" }
```

the remaining " , slkfjsdlfkj" is ignored

- If request document is

```
{ "aa" : "1" , "aa" : "2" }
```

then first "aa" would be used and second one would be ignored

Operation details

Asset (v1)

Asset types and subtypes possible combinations

type	subtype
datacenter	N_A
room	N_A
row	N_A
rack	N_A
group	input_power
	ANY_OTHER_FREE_FORM_STRING
device	epdu
	feed
	genset
	pdu
	rack controller
	router
	server
	sensor
	storage
	sts
	switch
	ups
	vm

Extended attributes consideration

Extended attributes belong to a list of attributes defined for an asset element to describe specific part of it.

There are two kinds of extended attributes :

- User extended attributes. Clients are allowed to add and update any couple of name and string value to complete the description of an asset element in a trusted way.
- Inventory device attributes. Specific device drivers are able to collect static data from devices and store them as extended attributes. A typical example is getting device serial number.

Considering that inventory device attributes are read only attributes, clients (webUI) should be aware about which attributes can be set. In consequence each extended attribute gets a "read_only" property.

Example 1 : inventory read only asset extended attribute

GET	/api/v1/asset/UPS01
Response : HTTP 200	


```
{
  "id": "1234",
  "name": "UPS01",
  "type": "device",
  "sub_type": "ups",
  "status": "active",
  "priority": "P1",
  "ext": [
    { "serial_no": "G214D17012", "read_only" : true }
  ]
}
```

Example 2 : user asset extended attribute (not read-only)

GET	/api/v1/asset/UPS02
Response : HTTP 200	
<pre>{ "id": "1235", "name": "UPS02", "type": "device", "sub_type": "ups", "status": "active", "priority": "P1", "ext": [{ "warranty_expiration_date": "2020-31-12", "read_only": false }] }</pre>	

export csv(Read)

Status	Operation	Method	URI	Description
Actual	Export	GET	/api/v1/asset/export	export all asset elements in a CSV file

Version : 1

Security : restricted access, with minimum security of level 2

Response document format in case of success : CSV UTF8 file

In Content-Disposition header, the file name is postfixed by the timestamp of the export (YYYY-MM-DD)

GET	/api/v1/asset/export
Response : HTTP 200	

```
Content-Disposition: attachment;
filename="asset_export2015-03-09.csv"
Content-Type: text/plain; charset=UTF-8
id,name,type,sub_type,description,location
1,DC-LAB2,datacenter,,
2,ROOM-02,room,,DC-LAB2
3,ANNEX-02,room,,DC-LAB2
4,CAGE-02,group,cage,,ROOM-02
5,ROW-03,row,,ROOM-02
6,ROW-04,row,,Ting Vit,ROOM-02
7,RACK-02,rack,,ROW-03
```

Response document in case of error: JSON doc

The error JSON document would contain the following information according the error message template:

Condition	Code
Internal error, problems with connection to database or database is inconsistent state. See log for more detail	42

import csv(Create)

Status	Operation	Method	URI	Description
Actual	Import	POST	/api/v1/asset/import	import CSV asset description file

Version : 1

Security : restricted access, with minimum security of level 2

Request document format : CSV UTF8, UTF16-LE, ASCII import file with the form field name **assets**

The HTTP headers to use when making an CSV import request include :

```
Content-Type: multipart/form-data; boundary="---bouNdary---"

Content-Length: Set to the number of bytes in entire request
body.

Authorization: Bearer your_auth_token
```

Response document format : JSON

- success

key	type	description
imported_lines	number	number of successfully imported lines
errors	array	array of arrays
errors::item	array	contains information about the error in one row
errors::item::first	number	number of the row, that had some errors
errors::item::second	string	Message, that indicates, why row wasn't successfully reported

- error (see standard JSON error template)

Condition	Code
Method is not allowed	45
Content size is too big	53
File "assets" is missing	46
File "assets" has bad coding or bad format	47

Internal error (no connection to database, ...)	42
Mandatory columns are missing in the csv file	46
Load csv was success, but error occurred during configuration sending of asset change notification. Consult system log.	42
Request document has invalid syntax. Cannot detect the delimiter, use comma (,) semicolon (;) or tabulator	48

Example 1 : UTF8 asset CSV import

POST	/api/v1/asset/import
<pre>Content-Type: multipart/form-data; boundary="---bouNdaRY---" Content-Length: 574 Authorization: GkH5MS0FxV/IsMJhP/QqL0wiCHMRZByR5MUPyEoHHeK= ---bouNdaRY--- Content-Disposition: file; assets="BIOS_asset_UTF8.csv" Content-Type: text/plain; charset=UTF-8 name;type;sub_type;description;location DC-LAB;datacenter;; ROOM-01;room;;DC-LAB ANNEX-01;room;;DC-LAB CAGE-01;group;cage;;ROOM-01 ROW-01;row;;ROOM-01 ROW-02;row;;Ting Vit;ROOM-01 RACK-01;rack;;ROW-01 RACK-01;rack;;ROW-02 ---bouNdaRY---</pre>	
Response : HTTP 200	
<pre>{ "imported_lines": 7, "errors" : [[8, "Name RACK-01 is already used"]] }</pre>	

The curl equivalent command is

```
curl -i -H "Authorization: Bearer GkH5MS0FxV/IsMJhP/QqL0wiCHMRZByR5MUPyEoHHeK=" -H "Expect:" -F
"assets=@./BIOS_asset_UTF8.csv;type=text/plain;charset=UTF-8" /api/v1/asset/import
```

Example 2 : UTF16-LE asset CSV import.

POST	/api/v1/asset/import
<pre>Content-Type: multipart/form-data; boundary="---bouNdaRY---" Content-Length: 574 Authorization: GkH5MS0FxV/IsMJhP/QqL0wiCHMRZByR5MUPyEoHHeK= ---bouNdaRY--- Content-Disposition: file; assets="BIOS_asset_UTF16.csv" Content-Type: text/plain; charset=UTF-16 name;type;sub_type;description;location; DC-LAB2;datacenter;;; ROOM-02;room;;DC-LAB2; ANNEX-02;room;;DC-LAB2; CAGE-02;group;cage;;ROOM-02; ROW-03;row;;ROOM-02; ROW-04;row;;Ting Vit;ROOM-02; RACK-02;rack;;ROW-03; ---bouNdaRY---</pre>	

Response : HTTP 200

```
{
  "imported_lines": 7,
  "errors" : []
}
```

The curl equivalent command is

```
curl -i -H "Authorization: Bearer GkH5MS0FxV/IsMJhP/QqL0wiCHMRZByR5MUPyEoHHeK=" -H "Expect:" -F
"assets=@./BIOS_asset_UTF16.csv;type=text/plain;charset=UTF-16" /api/v1/asset/import
```

Create one asset element

It is a general call for creating one asset of any type

Status	Operation	Method	URI	Description
Actual	Create	POST	/api/v1/asset	create a new asset element

Comment:

- **Attention! Format of response document for the read operation is not the same as formation of request document in post operation**
- **Not all ext attributes are listed here. Take a look on appropriate paragraph of read operations.** See also [RFC-18: List of known extended attributes](#) for attributes that are syntactically not-required, but have a special meaning in certain cases.

Restrictions:

- no functional for groups is implemented
- no functional for power chain is implemented
- tested only for type='datacenter' and 'device'

Version : 1

Security : restricted access, with minimum security of level 2

Request document format : JSON

Key	Type	Mandatory	Description
name	string(50)	Yes	name of the asset
type	string(50)	Yes	List of possible values: see section "Asset types and subtypes possible combination"
sub_type	string(50)	Yes	This field is not empty only if type is "device", otherwise it must present but should be empty List of possible values: see section "Asset types and subtypes possible combination"
status	string	Yes	List of possible values: {active, nonactive, spare, retired}
priority	string	Yes	List of possible values: {P1, P2, P3, P4, P5}
location	string	Yes	The name of the parent asset element. If element is unlocated or the element is a datacenter then the field should present and be empty.
powers	array	No	A list of direct power sources. Only for devices. Can be empty.
powers::Object::src_id	string	No	id of the asset (just to be consistent with GET method)
powers::Object::src_name	string	Yes	name of the asset (according this information identification of the asset is done)
powers::Object::src_socket	string	No	identification label of the socket on the source device
powers::Object::dest_socket	string	No	identification label of the socket on the device itself
groups	array	No	A list of group the asset belongs to. Can be empty
groups::Object::name	string	Yes	Name of the group
groups::Object::id	string	No	asset id of the group
ips	array	No	List of IP addresses

ips::item	string		string representation of IP
hostnames	array	No	List of hostnames. For the moment we agree to have only one item.
hostnames::item	string		string representation of hostname
macs	array	No	List of mac addresses
macs::item	string		string representation of mac address
fqdns	array	No	List of fqdn names
fqdns::item	string		string representation of fqdn name
ext	array	No	optional list of extended attributes that the user is free to fill. For a list of known strings please see: RFC-18

Response document format : JSON

- success

key	type	description
id	String	new asset identifier

- error (see standard JSON error template)

Condition	Code
Request document have wrong format or error in the syntax	48
Request document is empty	48
Request document has key "id"	51
Request document key "type" has unsupported value	47
Request document doesn't contain all required data	46
Internal error.	42
If parent, power source, or group specified doesn't exists in database	44
serial_no is not unique	47
Asset was created, but error occurred during configuration sending of asset change notification. Consult system log.	42

Example 1 : new datacenter

POST	/api/v1/asset
<pre>{ "name" : "Mydc", "type" : "datacenter", "sub_type" : "", "status" : "active", "priority" : "P1", "ext":{ "description" : "My Datacenter", "asset_tag" : "100ADi", "address": "Prague, Czech Republic" } }</pre>	
Response : HTTP 200	
<pre>{ "id": "1234" }</pre>	

Example 2 : new device

POST	/api/v1/asset
<pre>{ "name" : "the_first_device", "type" : "device", "sub_type" : "server", "status" : "active", "priority" : "P5", "location" : "the_zero_server", "ext":{ "hostname.1": "aaa", "fqdn.1": "aaa.bb.ccc.com" } }</pre>	
Response : HTTP 200	
<pre>{ "id": "1235" }</pre>	

Delete one asset element

Status	Operation	Method	URI	Description
Actual	Delete	DELETE	/api/v1/asset/<asset-id>	Delete requested asset

The asset may be deleted only if there is no child dependences.

Version : 1

Security : restricted access, with minimum security of level 2

Request parameters :

- **asset-id** : specific asset id.

Response document format : JSON

- success

JSON document	{}
---------------	----

- error (see standard JSON error template)

Condition	Code
Parameter id is missing	46
Element not found	44
Delete is not possible as asset has elements inside	50
Asset was deleted, but error occurred during configuration sending of asset change notification. Consult system log.	42

Example : a delete request on an asset with no dependences.

DELETE	/api/v1/asset/1234
{}	
Response : HTTP 200	

Update one asset element

It is a general call for updating one asset of any type.

Limitation:

- As input WS should receive the whole information about an asset (uses the same JSON request document as for create operation).

Status	Operation	Method	URI	Description
Actual	Update	PUT	/api/v1/asset/<asset-id>	Update an existing asset element

Comment:

- Request document has the same JSON structure as request document in "Create one asset element "**
- Attention! Format of response document for the read operation is not the same as formation of request document in put operation**
- Not all ext attributes are listed here. Take a look on appropriate paragraph of read operations.**

Restrictions:

- no functional for groups is implemented
- no functional for power chain is implemented
- tested only for type='device' with limitations and for type='datacenter'

Version : 1

Security : restricted access, with minimum security of level 2

Request document format : JSON

Response document format : JSON

- success

key	type	description
id	String	asset identifier

- error (see standard JSON error template)

Condition	Code
Request document have wrong format or error in the syntax	48
Request document is empty	48
Request document have keys that are not implemented yet	48
Request document has key "id"	51
Request document key "type" has unsupported value	47
Request document doesn't contain all required data	46
Internal error.	42
More than 2 epdu/pdu are places in the rack	51
If parent, power source, or group specified doesn't exists in database	44
serial_no is not unique	47
Asset was updated, but error occurred during configuration sending of asset change notification. Consult system log.	42

Example 1 : update a datacenter

PUT	/api/v1/asset/12345
-----	---------------------

<pre>{ "name" : "Mydevice", "type" : "datacenter", "sub_type" : "N_A", "status" : "active", "priority" : "P1", "ext":{ "description" : "My Datacenter", "asset_tag" : "100ADi", "address": "Prague, Czech Republic" } }</pre>
Response : HTTP 200
<pre>{ "id": "12345" }</pre>

Example 2 : update a device

PUT	/api/v1/asset/12345
<pre>{ "name" : "Mydevice", "type" : "device", "sub_type" : "server", "status" : "active", "priority" : "P1", "location" : "MyDc", "ext":{ "description" : "My device in Datacenter", "asset_tag" : "100ADi" } }</pre>	
Response : HTTP 200	
<pre>{ "id": "12345" }</pre>	

assets (Read)

version	Operation	Method	URI	Description
Actual	Read	GET	/api/v1/assets?in=<container_id>&type=type1[,type2[,...]]	Retrieves the list of assets of specified type/s and placed into the container. List of possible "type" values: see section "Asset types and subtypes possible combination"
Actual	Read	GET	/api/v1/assets?in=<container_id>&sub_type=subtype1[, subtype2[,...]]	Retrieves the list of assets of specified subtype/s, specified subtype and placed into the specified container. List of possible :subtype" values: see section "Asset types and subtypes possible combination"

Version : 1

Security : Public access (level 0)

Response document format : JSON

Property name	type	mandatory	description
[]	array	Yes	array of assets that match requested criteria
assets/id	String	Yes	id of each asset element
assets/name	string(50)	Yes	name of each asset element
assets/type	string(50)	Yes	type of each asset element
assets/sub_type	string(50)	Yes	subtype of each asset element

Example :

GET	/api/v1/assets?type=room&in=1
Response : HTTP 200	
<pre>[{ "id": "12345", "name": "my first room", "type": "room", "sub_type": "N_A" }, { "id": "123456", "name": "my second room", "type": "room", "sub_type": "N_A" }]</pre>	

GET	/api/v1/assets?type='DNA'
Response : HTTP 400	

GET	/api/v1/assets?sub_type=epdu,ups&in=1
Response : HTTP 200	
<pre>[{ "id": "12345", "name": "my first ups", "type": "device", "sub_type": "ups" }, { "id": "123456", "name": "my epdu", "type": "device", "sub_type": "epdu" }]</pre>	

devices, datacenters, rooms, racks, rows (Read)

Operation	Method	URI	Description
Read	GET	/api/v1/asset/devices?[subtype="subtype_name",...]	Retrieves the list of all devices regardless the datacenter
Read	GET	/api/v1/asset/datacenters	Retrieves the list of datacenters regardless datacenter
Read	GET	/api/v1/asset/rooms	Retrieves the list of rooms regardless the datacenter
Read	GET	/api/v1/asset/racks	Retrieves the list of racks regardless the datacenter
Read	GET	/api/v1/asset/rows	Retrieves the list of rows regardless the datacenter

Version : 1

Security : Public access (level 0)

Request parameters:

parameter	type	mandatory	description	example
subtype	string	No	This is list of device subtype we are interested in. If parameter is not specified return all devices. List of possible values: see section "Asset types and subtypes possible combination"	subtype=ups subtype=ups,pdu

Response document format : JSON

Example :

GET	/api/v1/asset/devices
Response : HTTP 200	
<pre>{ "devices": [{ "id": "1234", "name": "srv1" }, { "id": "1234", "name": "srv1" }] }</pre>	

GET	/api/v1/asset/datacenters
Response : HTTP 200	

```
{
  "datacenters": [
    {
      "id": "1234",
      "name": "DC4"
    },
    {
      "id": "1234",
      "name": "DC1"
    }
  ]
}
```

GET	/api/v1/asset/rows
-----	--------------------

Response : HTTP 200

```
{
  "rows": [
    {
      "id": "1234",
      "name": "row1"
    },
    {
      "id": "1234",
      "name": "row119"
    }
  ]
}
```

GET	/api/v1/asset/racks
-----	---------------------

Response : HTTP 200

```
{
  "racks": [
    {
      "id": "1234",
      "name": "rack in the lab"
    },
    {
      "id": "1234",
      "name": "one more rack in the lab"
    }
  ]
}
```

GET	/api/v1/asset/rooms
Response : HTTP 200	
<pre>{ "rooms": [{ "id": "1234", "name": "room second floor" }, { "id": "1234", "name": "room third floor" }] }</pre>	

datacenter (Read)

Status	Operation	Method	URI	Description
Actual	Read	GET	/api/v1/asset/<dc-id>	Retrieves information about a requested datacenter
Deprecated	Read	GET	/api/v1/asset/datacenter/<dc-id>	Retrieves information about a requested datacenter

Version : 1

Security : Public access (level 0)

Request Parameters :

- **dc-id** : specific datacenter id.

Response Document : JSON doc

Key	Type	Mandatory	Description
id	String	Yes	dc identifier
name	string(50)	Yes	dc name

type	string	Yes	"datacenter"
sub_type	string	Yes	"N_A"
status	string	Yes	List of possible values: {active, nonactive, spare, retired}
location	String	Yes	Empty string, as datacenter has no parent
priority	string	Yes	List of possible values: {P1, P2, P3, P4, P5}
power_devices_in_uri	string	Yes	A relative URI to select all power devices in the datacenter (all 4 levels down) Currently: "/api/v1/assets?in=<dc_ID>&sub_type=epdu,pdu,feed,genset,ups"
groups	array	No	A list of group the asset belongs to. Can be empty
groups::Object::name	string	Yes	Name of the group
groups::Object::id	string	No	asset id of the group
ext	array	No	optional list of extended attributes that's the user is free to fill it.
parents	array	Yes	Empty list for datacenter
predefined ext key			description
ext::asset_tag	string	No	asset tag
ext::description	string(255)	No	Description
ext::total_facility_load	number	No	Nominal facility load (in kVA)
ext::company	string	No	Company name
ext::site_name	string	No	Site name
ext::region	string	No	region name
ext::country	string	No	country
ext::address	string	No	Address
ext::contact_name	string	No	Contact name
ext::contact_email	string	No	Contact mail
ext::contact_phone	string	No	Contact phone

Example :

GET	/api/v1/asset/DC1234
Response : HTTP 200	
<pre>{ "id": "DC1234", "name": "dcl", "type": "datacenter", "sub_type": "N_A", "status": "active", "priority": "P1", "ext": [{"total_facility_load": 10, "read_only" : false}, {"contact_name": "manager@dc.com", "read_only" : false}] }</pre>	

room(Read)

Status	Operation	Method	URI	Description
Actual	Read	GET	/api/v1/asset/<room-id>	Retrieves information about a requested room
Deprecated	Read	GET	api/v1/asset/room/<room-id>	Retrieves information about a requested room

Version : 1

Security : Public access (level 0)

Request Parameters :

- **room-id** : specific room id.

Response Document : JSON doc

Key	Type	Mandatory	Description
id	String	Yes	room identifier
name	string(50)	Yes	room name
type	string	Yes	"room"
sub_type	string	Yes	"N_A"
status	string	Yes	List of possible values: {active, nonactive, spare, retired}
priority	string	Yes	List of possible values: {P1, P2, P3, P4, P5}
location	String	Yes	Parent name. If asset, is not located (so it doesn't have a parent) this field is still returned, but it is an empty string.
power_devices_in_uri	string	Yes	A relative URI to select all power devices in the room (all 3 levels down) Currently: "/api/v1/assets?in=<room_ID>&sub_type=epdu,pdu,feed,genset,ups"
location_id	String	No	parent id
location_uri	URI	No	optional location of the room. nothing means it is not currently located.
groups	array	No	A list of group the asset belongs to. Can be empty
groups::Object::name	string	Yes	Name of the group
groups::Object::id	string	No	asset id of the group
ext	array	No	optional list of extended attributes that's the user is free to fill it.
parents	array	Yes	List of parents, sorted from nearest element
parents::N::name	string	Yes	name of parent N
parents::N::id	number	Yes	id of parent N
parents::N::type	string	Yes	type of parent N
parents::N::sub_type	string	Yes	sub_type of parent N
predefined ext key			description
ext::asset_tag	string(255)	No	asset tag
ext::description	string(255)	No	Description
ext::contact_name	string(255)	No	Contact name
ext::contact_email	string(255)	No	Contact mail
ext::contact_phone	string(255)	No	Contact phone

Example :

GET	/api/v1/asset/R1234
Response : HTTP 200	

```
{
  "id": "R1234",
  "name": "room1",
  "type": "room",
  "sub_type": "N_A",
  "priority": "P2",
  "status": "active",
  "location_uri": "/api/v1/asset/3631234"
}
```

row(Read)

Status	Operation	Method	URI	Description
Actual	Read	GET	/api/v1/asset/<row-id>	Retrieves information about a requested row
Deprecated	Read	GET	/api/v1/asset/row/<row-id>	Retrieves information about a requested row

Version : 1

Security : Public access (level 0)

Request Parameters :

- **row-id** : specific row id.

Response Document : JSON doc

Key	Type	Mandatory	Description
id	String	Yes	row identifier
status	string	Yes	List of possible values: {active, nonactive, spare, retired}
type	string	Yes	"row"
sub_type	string	Yes	"N_A"
priority	string	Yes	List of possible values: {P1, P2, P3, P4, P5}
name	string(50)	Yes	row name
location	String	Yes	Parent name. If asset, is not located (so it doesn't have a parent) this field is still returned, but it is an empty string.
power_devices_in_uri	string	Yes	A relative URI to select all power devices in the row (all 2 levels down) Currently: "/api/v1/assets?in=<row_ID>&sub_type=epdu,pdu,feed,genset,ups"
location_id	String	No	parent id
location_uri	URI	No	optional location of the row. Nothing means it is not currently located.
groups	array	No	A list of group the asset belongs to. Can be empty
groups::Object::name	string	Yes	Name of the group
groups::Object::id	string	No	asset id of the group
ext	array	No	optional list of extended attributes that's the user is free to fill it.
parents	array	Yes	List of parents, sorted from nearest element
parents::N::name	string	Yes	name of parent N
parents::N::id	number	Yes	id of parent N
parents::N::type	string	Yes	type of parent N
parents::N::sub_type	string	Yes	sub_type of parent N

predefined ext key			description
ext::asset_tag	string	No	asset_tag
ext::description	string(255)	No	Description
ext::maximum_number_racks	number	No	rack capacity
ext::contact_name	string	No	Contact name
ext::contact_email	string	No	Contact mail
ext::contact_phone	string	No	Contact phone

Example :

GET	/api/v1/asset/RW1234
Response : HTTP 200	
<pre>{ "id": "RW1234", "name": "row1", "type": "row", "sub_type": "N_A", "status": "active", "priority": "P1", "location_uri": "/api/v1/asset/221234" }</pre>	

groups (Read)

Status	Operation	Method	URI	Description
Actual	Read	GET	/api/v1/asset/groups	Retrieves the list of groups

Version : 1

Security : Public access (level 0)

Response Document : JSON

Key	Type	Mandatory	Description
groups[]	array	Yes	an array of groups
groups::id	string	Yes	id of each group
groups::name	string(50)	Yes	name of each group

Example :

GET	/api/v1/asset/groups
Response : HTTP 200	


```
{
  "groups": [
    {
      "id": "1234",
      "name": "group1"
    }
  ]
}
```

group(Read), (Create, Update) Not Yet Implemented

Status	Operation	Method	URI	Description
Actual	Read	GET	/api/v1/asset/<group-id>	Retrieves information about a requested group
Deprecated	Read	GET	/api/v1/asset/group/<group-id>	Retrieves information about a requested group

Version : 1

Security : Public access (level 0)

Request Parameters :

- **group-id** : specific group id.

Response Document : JSON doc

Key	Type	Mandatory	Description
id	String	Yes	group identifier
status	string	Yes	List of possible values: {active, nonactive, spare, retired}
priority	string	Yes	List of possible values: {P1, P2, P3, P4, P5}
name	string(50)	Yes	group name
type	string	Yes	"group"
sub_type	string	Yes	List of possible values: {"input power"}
location	String	Yes	Parent name. If asset, is not located (so it doesn't have a parent) this field is still returned, but it is an empty string.
location_id	String	No	parent id
location_uri	URI	No	optional location of the group. nothing means it is not currently located.
ext	array	No	optional list of extended attributes that's the user is free to fill it.
parents	array	Yes	List of parents, sorted from nearest element
parents::N::name	string	Yes	name of parent N
parents::N::id	number	Yes	id of parent N
parents::N::type	string	Yes	type of parent N
parents::N::sub_type	string	Yes	sub_type of parent N
predefined ext key			description
ext::description	string(255)	No	Description
ext::contact_name	string(255)	No	Contact name
ext::contact_email	string(255)	No	Contact mail
ext::contact_phone	string(255)	No	Contact phone

Example :

GET	/api/v1/asset/21234
------------	----------------------------

Response : HTTP 200

```
{
  "id": "21234",
  "name": "cage1234",
  "type": "group",
  "sub_type": "input_power",
  "status": "active",
  "priority": "P2",
  "location_uri": "/api/v1/asset/1234"
}
```

status	Operation	Method	URI	Description
Not implemented	Create	POST	/api/v1/asset/group/	create a new group

Version : 1

Security : restricted access, with minimum security of level 2

Request Document : JSON doc

Key	Type	Mandatory	Description
name	string(50)	Yes	group name
status	string	Yes	List of possible values: {active, nonactive, spare, retired}
priority	string	Yes	List of possible values: {P1, P2, P3, P4, P5}
type	string	Yes	"group"
sub_type	string	Yes	e.g. "input_power"
location	String	Yes	Parent name. If asset, is not located (so it doesn't have a parent) this field is still returned, but it is an empty string.
location_id	String	No	parent id
location_uri	URI	No	optional location of the group. nothing means it is not currently located.
ext	array	No	optional list of extended attributes that's the user is free to fill it.
parents	array	Yes	List of parents, sorted from nearest element
parents::N::name	string	Yes	name of parent N
parents::N::id	number	Yes	id of parent N
parents::N::type	string	Yes	type of parent N
parents::N::sub_type	string	Yes	sub_type of parent N
predefined ext key			Description
ext::description	string(255)	No	Description
ext::contact_name	string(255)	No	Contact name
ext::contact_email	string(255)	No	Contact mail
ext::contact_phone	string(255)	No	Contact phone

Response Document : JSON doc

Property name	type	description
---------------	------	-------------

id	String	new group identifier
----	--------	----------------------

Example :

POST	/api/v1/asset/group
<pre>{ "name": "dcInputPowerChain", "status": "active", "priority": "P5", "type": "group", "sub_type": "input_power", "location_uri": "/api/v1/asset/1234", "ext": { "description": "this group manages the input power chain" } }</pre>	
Response : HTTP 200	
<pre>{ "id": "1234" }</pre>	

Status	Operation	Method	URI	Description
Not implemented	Update	PUT	/api/v1/asset/<group-id>	update a requested group resource

Version : 1

Security : restricted access, with minimum security of level 2

Request Parameters :

- **group-id** : specific group id.

Request Document : JSON doc

The same format like POST Request document.

Example :

PUT	/api/v1/asset/1234
<pre>{ "name": "dcInputPowerChainUpdated", "type": "group", "status": "active", "priority": "P3", "sub_type": "input_power", "location_uri": "/api/v1/asset/1234", "ext": { "description": "this group manages the input power chain (updated)" } }</pre>	

Response : HTTP 200

rack(Read)

Status	Operation	Method	URI	Description
Actual	Read	GET	/api/v1/asset/<rack-id>	Retrieves information about a requested rack
Deprecated	Read	GET	/api/v1/asset/rack/<rack-id>	Retrieves information about a requested rack

Version : 1

Security : Public access (level 0)

Request Parameters :

- **rack-id** : specific rack id.

Response Document : JSON doc

Key	Type	Mandatory	Description
id	String	Yes	rack identifier
status	string	Yes	List of possible values: {active, nonactive, spare, retired}
type	string	Yes	"rack"
sub_type	string	Yes	"N_A"
priority	string	Yes	List of possible values: {P1, P2, P3, P4, P5}
name	string(50)	Yes	rack name
location	String	Yes	Parent name. If asset, is not located (so it doesn't have a parent) this field is still returned, but it is an empty string.
power_devices_in_uri	string	Yes	A relative URI to select all power devices in the rack Currently: "/api/v1/assets?in=<rack_ID>&sub_type=epdu,pdu,feed,genset,ups"
location_id	String	No	parent id
location_uri	URI	No	optional location of the rack. Nothing means it is not currently located.
groups	array	No	A list of group the asset belongs to. Can be empty
groups::Object::name	string	Yes	Name of the group
groups::Object::id	string	No	asset id of the group
ext	array	No	optional list of extended attributes that's the user is free to fill it.
parents	array	Yes	List of parents, sorted from nearest element
parents::N::name	string	Yes	name of parent N
parents::N::id	number	Yes	id of parent N
parents::N::type	string	Yes	type of parent N
parents::N::sub_type	string	Yes	sub_type of parent N
Predefined ext key			Description
ext::description	string(255)	No	Description
ext::model	string(255)	No	Model type
ext::manufacturer	string(255)	No	Company producing the rack
ext::u_size	number	No	number of total U available
ext::asset_tag	string(255)	No	Asset tag
ext::serial_no	string(255)	No	Serial number
ext::contact_name	string(255)	No	Contact name
ext::contact_email	string(255)	No	Contact mail

ext::contact_phone	string(255)	No	Contact phone
Computed values			Description
computed::freeusize	number	Yes	Number of free U positions for the rack, null if not available
computed::outlet.available	Object	Yes	Information about available sockets, which is an object with pdu/epdu id as a key and number of available sockets as a value (null if not available). There is special key called sum , which contains summary of all values or null if some pdu/epdu returns null.
computed::realpower.nominal	number	Yes	Number of nominal power that rack can handle. Null if the value is not available.

Example :

GET	/api/v1/asset/1234
Response : HTTP 200	
<pre>{ "id": "204895", "name": "rack1", "type": "rack", "sub_type": "N_A", "status": "active", "priority": "P2", "ext": [{ "model": "RESSPU4210KB 600mm x 1070mm - 42U Rack", "read_only": false }, { "manufacturer": "EATON", "read_only": false }, { "u_size": "27", "read_only": false }, { "asset_tag": "CAB00001", "read_only": false }], "computed": { "freeusize": null, "outlet.available": { "pdu1": 4, "pdu2": null, "sum": null }, "realpower.nominal": 3456 } }</pre>	

device(Read)

Status	Operation	Method	URI	Description
Actual	Read	GET	/api/v1/asset/<device-id>	Retrieves information about a requested device

Deprecated	Read	GET	/api/v1/asset/device/<device-id>	Retrieves information about a requested device
------------	------	-----	----------------------------------	--

Comment:

Attention! Format of response document for the read operation is not the same as formation of request document in post operation

Version : 1

Security : Public access (level 0)

Request Parameters :

- **device-id** : specific device id.

Response Document : JSON doc

Property name	type		mandatory	description
id	String		Yes	device identifier
name	string(50)		Yes	device name
status	string		Yes	List of possible values: {active, nonactive, spare, retired}
priority	string		Yes	List of possible values: {P1, P2, P3, P4, P5}
type	string		Yes	"device"
sub_type	string		Yes	List of possible values: see section "Asset types and subtypes possible combination"
ips	array		No	List of IP addresses
ips::item	string			string representation of IP
hostnames	array		No	List of hostnames. For the moment we agree to have only one item.
hostnames::item	string			string representation of hostname
macs	array		No	List of mac addresses
macs::item	string			string representation of mac address
fqdns	array		No	List of fqdn names
fqdns::item	string			string representation of fqdn name
location_uri	String		No	location URI of the parent
location	String		Yes	Parent name. If asset, is not located (so it doesn't have a parent) this field is still returned, but it is an empty string.
location_id	String		No	parent id
groups[]	array		No	A list of group the asset belongs to. Can be empty
groups::Object::name	string		Yes	Name of the group
groups::Object::id	string		No	asset id of the group
powers[]	array		No	optional array of inlet power link
powers::Object::src_socket	number		No	num of socket of the device
powers::Object::src_id	string		Yes	id of the source device
powers::Object::src_name	string		Yes	name of the source device
powers::Object::dest_socket	number		No	num of socket of the external device
outlets	Object		No	information about outlets (for pdu, epdu, ups). See description bellow.
ext	array		No	optional list of extended attributes that's the user is free to fill it.
parents	array		Yes	List of parents, sorted from nearest element
parents::N::name	string		Yes	name of parent N
parents::N::id	number		Yes	id of parent N
parents::N::type	string		Yes	type of parent N
parents::N::sub_type	string		Yes	sub_type of parent N

predefined ext key	type	read_only	mandatory	description
ext/description	string(255)	No	No	Description
ext/model	string	No	No	model description
ext/u_size	number	No	No	u height affected to the device (device height + extra space)
ext/location_u_pos	number	No	No	u position of device (down-up) 1UR means 1U Rear 1U means 1U front 1UF means 1U front
ext/location_w_pos	string	No	No	width position
ext/location_d_pos	front, rear	No	No	depth position
ext/location_front_pos	torX, sorX, borX	No	No	front position among : <ul style="list-style-type: none">• tor{1..4} : top of the rack• sor{1..4} : side of rack• bor{1..4} : bottom of rack
ext/location_rear_pos	torX, sorX, borX	No	No	rear position among : <ul style="list-style-type: none">• tor{1..4} : top of the rack• sor{1..4} : side of rack• bor{1..4} : bottom of rack
ext/model	string	No	No	model
ext/serial_no	string	No	No	serial number
ext/part_number	string	No	No	part number
ext/manufacture	string(255)	No	No	Company producing the device
ext/installation_date	date	No	No	installation date
ext/end_warranty_date	date	No	No	end initial warranty date
ext/warranty_expiration_date	date	No	No	warranty expiration date (with extension warranty)
ext/maintenance_date	date	No	No	maintenance date
ext/asset_tag	string	No	No	Asset tag
ext/contact_name	string	No	No	Contact name
ext/contact_email	string	No	No	Contact mail
ext/contact_phone	string	No	No	Contact phone
predefined "ups" ext key	type	read_only	mandatory	description
ext/model	string	Yes	Yes	model
ext/manufacture	string	Yes	Yes	Company producing the device
ext/serial_no	string	Yes	Yes	serial number
ext/ups.status	string	Yes	Yes	device internal status
ext/device.description	string	Yes	No	device internal description
ext/device.contact	string	Yes	No	device internal contact
ext/device.location	string	Yes	No	device internal location
ext/installation_date	date	No	No	installation date
ext/maintenance_date	date	No	No	Maintenance date
ext/end_warranty_date	date	No	No	End warranty date
ext/battery.type	string	No	No	Battery chemistry
ext/battery_installation_date	date	No	No	Battery installation date
ext/battery_warranty_expiration_date	date	No	No	Battery warranty expiration date (with extension warranty)
ext/battery_maintenance_date	date	No	No	Battery maintenance date

ext/asset_tag	string	No	No	Asset tag
ext/phases.output	number	Yes	Yes	Number of output phases
ext/phases.input	number	Yes	Yes	Number of input phases
predefined "epdu" ext key	type	read_only	mandatory	description
ext/model	string	Yes	Yes	model
ext/manufacturer	string	Yes	Yes	Company producing the device
ext/serial_no	string	Yes	Yes	serial number
ext/type	string	Yes	Yes	epdu
ext/outlet.count	number	Yes	Yes	Number of outlet
ext/device.description	string	Yes	No	device internal description
ext/device.contact	string	Yes	No	device internal contact
ext/device.location	string	Yes	No	device internal location
ext/installation_date	date	No	No	installation date
ext/maintenance_date	date	No	No	Maintenance date
ext/end_warranty_date	date	No	No	End warranty date
ext/asset_tag	string	No	No	Asset tag
ext/phases.output	number	Yes	Yes	Number of output phases
ext/phases.input	number	Yes	Yes	Number of input phases
ext/daisy_chain	number	No	No	daisy chain pdu support. 1 if for the master, 2 is for first slave, N is for the N-1 slave
predefined "genset" ext key	type	read_only	mandatory	description
ext/model	string	No	No	model
ext/runtime	number	No	Yes	autonomy duration in hour
ext/asset_tag	string	No	No	Asset tag
ext/serial_no	string	No	No	Serial number
ext/installation_date	date	No	No	Installation date
ext/maintenance_date	date	No	No	Maintenance date
ext/maintenance_due	date	No	No	Maintenance due date
ext/service_contact_name	string	No	No	Service contact name
ext/service_contact_mail	mail	No	No	Service contact name
ext/service_contact_phone	phone	No	No	Service contact phone
predefined "feed" ext key	type	read_only	mandatory	description
ext/phases.output	number	Yes	Yes	Number of output phases
ext/phases.input	number	Yes	Yes	Number of input phases

Outlets: it is an object, where each key is a string ("1", "2", "3" ...) a represents an outlet and each value is a an array of objects which are described below:

Key name	Value type	Value	Csv representation
name	string	Possible values: "label", "type", "group"	Column names: outlet.N.label / outlet.N.type / outlet.N.group, where N is a number of outlet
value	string	Some string	values for each row
read_only	boolean	true/false	ignored

Example :

GET	/api/v1/asset/1234
Response : HTTP 200	
<pre>{ "id": "1234", "name": "srv1", "type": "device", "sub_type": "server", "hostname": "srv1", "ip": "192.168.0.1", "ext": { { "manufacturer" : "HP", "read_only": false } } }</pre>	

Topology (v1)

location (Read)

Operation	Method	URI
Read	GET	/api/v1/topology/location?[[from=<location_id none>&recursive=true false&filter=<element_kind> feed_by=<source_id>]][to=<element_id

Version : 1

Security : Public access (level 0)

Description: this call returns the location topology to specified element or container (datacenter, row, ... , etc).

Request Parameters :

- **from** : location id or none of the starting point of the topology. none means for unlocated element.
- **to**: location uri of the ending point of the topology.
- **recursive** : if "true", do a recursive walk on the location of all child location element to get the full tree location topology. If parameter is empty or wasn't specified then it is treated as "false"
- **feed_by**: provide an opportunity to bring power topology restrictions into location topology. If this parameter is specified and it is XYZ, then call would return all devices that have XYZ in source power chain. If this parameter is specified, than parameter "filter" must have value "devices". Using the from=none together with feed_by is forbidden.
- **filter** : combined with a recursive walk, the filter allows to return one kind of element. Allowed filter element are :
 - rooms
 - rows
 - racks
 - devices
 - groups

Response Document : JSON doc

Key	Type	Mandatory	Description
name	string(50)	Yes	description
id	string	Yes	element identifier
contains{}	object	Yes	list of objects "rooms", "rows", "racks", "devices", "groups"

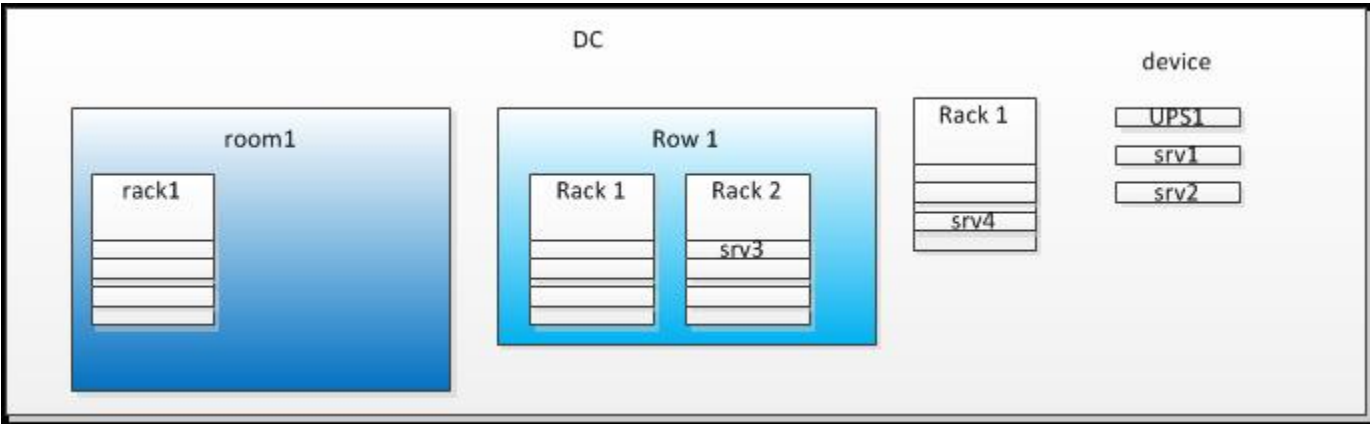
contains::rooms[]	array	No	array of rooms
contains::rooms::name	string(50)	Yes	name
contains::rooms::id	string	Yes	identifier
contains::rooms::type	string	Yes	The type of the asset
contains::rooms::sub_type	string	Yes	The sub_type of the asset
contains::rooms::contains{}	object	Yes	list of objects "rows", "racks", "devices", "groups"
contains::rows[]	array	No	array of rows
contains::rows::name	string(50)	Yes	name
contains::rows::id	string	Yes	identifier
contains::rows::type	string	Yes	The type of the asset
contains::rows::sub_type	string	Yes	The sub_type of the asset
contains::rows::contains{}	object	Yes	list of objects "racks", "devices", "groups"
contains::racks[]	array	No	array of racks
contains::racks::name	string(50)	Yes	name
contains::racks::id	string	Yes	identifier
contains::racks::type	string	Yes	The type of the asset
contains::racks::sub_type	string	Yes	The sub_type of the asset
contains::racks::contains{}	object	Yes	list of objects "devices", "groups"
contains::devices[]	array	No	array of devices
contains::devices::name	string(50)	Yes	name
contains::devices::id	string	Yes	identifier
contains::devices::type	string	Yes	The type of the asset
contains::devices::sub_type	string	Yes	The sub_type of the asset
contains::devices::contains{}	object	Yes	list of objects "devices", "groups"
contains::groups[]	array	No	array of groups
contains::groups::name	string(50)	Yes	name
contains::groups::id	string	Yes	identifier
contains::groups::type	string	Yes	The type of the asset
contains::groups::sub_type	string	Yes	The sub_type of the group
contains::groups::contains{}	object	Yes	list of objects "rooms", "rows", "racks", "devices"

Error states: JSON doc

Condition	error code	message
both parameters 'from' and 'to' are non-empty	52	Only one parameter can be specified at once: 'from' or 'to'.
both parameters 'from' and 'to' are empty	46	Parameter 'from/to' is required.
element of 'to' or 'from' or 'feed_by' is unknown	44	unknown element (not found)
parameter 'recursive' has bad value	47	Parameter 'recursive' has bad value. Received %. Expected 'true'/'false'
parameter 'filter' has bad value	47	Parameter 'filter' has bad value. Received %. Expected 'groups'/'devices'/'rooms'/'rows'/'racks'.
wrong filter used for request with filter	52	Request cannot be processed because of conflict in parameters. With variable 'feed_by' can be specified only 'filter=devices'.

request for unlocated elements feeded by device	52	Request cannot be processed because of conflict in parameters. With variable 'feed_by' variable 'from' can not be 'none'
--	----	--

Examples : considering a DC composed of several kinds of elements illustrated bellow :



Example 1 : a first level DC location topology request

GET	/api/v1/topology/location?from=DC1234
Response : HTTP 200	
<pre>{ { "name": "dc1", "id": "DC1234", "type": "datacenter", "sub_type": "N_A", "contains": { "rooms": [{ "name": "room1", "id": "R01234", "type": "room", "sub_type": "N_A" }], "rows": [{ "name": "row1", "id": "RW1234", "type": "row", "sub_type": "N_A" }], "racks": [{ "name": "Rack1", "id": "RCK1234", "type": "rack", "sub_type": "N_A" }], "devices": [{ "name": "UPS1", "id": "UP1234", "type": "datacenter", "sub_type": "ups" }, { "name": "srv1", "id": "SRV1234", "type": "datacenter", "sub_type": "server" }, { "name": "srv2", "id": "SRV1235", "type": "datacenter", "sub_type": "server" }] } } }</pre>	

Example 2 : a whole recursive DC location topology request

GET	/api/v1/topology/location?from=DC1234&recursive=true
Response : HTTP 200	

```

{
  {"name":"dc1","id" : "DC1234", "type":"datacenter", "sub_type":"N_A",
  "contains": {
    "rooms":[{"name":"room1","id":"R01234", "type":"room", "sub_type":"N_A",
    "contains":{"racks":[{"name":"rack1","id":"RCK12345", "type":"rack", "sub_type":"N_A"}]}
  ]},
  "rows":[{"name":"row1","id":"RW1234", "type":"row", "sub_type":"N_A",
  "contains":{
    "racks":[
      {"name":"Rack1","id":"RCK12346", "type":"rack", "sub_type":"N_A"},
      {"name":"Rack2","id":"RCK12347", "type":"rack", "sub_type":"N_A",
      "contains":{"devices":[{"name":"srv3","id":"SRV1236", "type":"device", "sub_type":"server"}]}
    ]}
  ]},
  "racks":[{"name":"Rack1","id":"RCK1234", "type":"rack", "sub_type":"N_A"
  "contains":{"devices":[{"name","srv4","id":"SRV1237", "type":"device", "sub_type":"server"}]}
  ]},
  "devices":[
    {"name":"UPS1", "id":"UP1234", "type":"device", "sub_type":"ups"},
    {"name":"srv1", "id":"SRV1234", "type":"device", "sub_type":"server"},
    {"name":"srv2", "id":"SRV1235", "type":"device", "sub_type":"server"}]
  }
}

```

Example 3 : a recursive DC location topology filtered by racks element type

GET	/api/v1/topology/location?from=DC1234&recursive=true&filter=racks
Response : HTTP 200	

```
{
  {
    "name": "dc1", "id" : "DC1234", "type": "datacenter", "sub_type": "N_A",
    "contains": {
      "rooms": [{
        "name": "room1", "id": "R01234", "type": "room", "sub_type": "N_A",
        "contains": {
          "racks": [{
            "name": "rack1", "id": "RCK12345", "type": "rack", "sub_type": "N_A"}]}
      ]},
      "rows": [{
        "name": "row1", "id": "RW1234", "type": "row", "sub_type": "N_A",
        "contains": {
          "racks": [
            {
              "name": "Rack1", "id": "RCK12346", "type": "rack", "sub_type": "N_A"},
            {
              "name": "Rack2", "id": "RCK12347", "type": "rack", "sub_type": "N_A"}]}
      ]},
      "racks": [{
        "name": "rack1", "id": "RCK1234", "type": "rack", "sub_type": "N_A"}],
    }
  }
}
```

Example 4 : request the location topology of server srv3 :

GET	/api/v1/topology/location?to=SRV1236
Response : HTTP 200	
<pre>{ { "name": "dc1", "id" : "DC1234", "type": "datacenter", "sub_type": "N_A", "contains": { "rows": [{ "name": "row1", "id": "RW1234", "type": "row", "sub_type": "N_A", "contains": { "racks": [{ "name": "Rack2", "id": "RCK12347", "type": "rack", "sub_type": "N_A", "contains": { "devices": [{ "name": "srv3", "id": "SRV1236", "type": "device", "sub_type": "server"}]} }]}]}, } } }</pre>	

Example 5 : request unlocated elements :

GET	/api/v1/topology/location?from=none
Response : HTTP 200	

```
{
  "racks": [{ "name": "rack99", "id": "RCK12399", "type": "rack", "sub_type": "N_A" }],
  "devices": [{ "name": "srv99", "id": "SRV1299", "type": "device",
    "sub_type": "server" } ]
}
```

Example 6 : request unlocated devices :

GET	/api/v1/topology/location?from=none&filter=devices
Response : HTTP 200	
<pre>{ "devices": [{ "name": "srv3", "id": "SRV1236", "type": "device", "sub_type": "server" }] }</pre>	

power (Read)

Operation	Method	URI	Description
Read	GET	/api/v1/topology/power?[filter_dc=<dc_id>][filter_group=<group_id>][to=<device_id>][from=<device_id>]	Retrieves the power topology from a specific location starting point

Version : 1

Security : Public access (level 0)

Request Parameters :

- **from, to or filter :**
 - **from=<device_id>** : returns the list of devices directly powered by (or connected) to the requested device (device_id).
 - **to=<device_id>** : returns the full power chain which powers the requested target device.
 - **filter_dc=<dc_id>** : returns all power links, where both source and destination elements are in the specified datacenter.
 - **filter_group=<group_id>** : returns all power links, where both source and destination element are in the specified group.

Response Document : JSON doc

Key	Type	Mandatory	Description
devices[]	array	Yes	array of devices
devices::name	string(50)	Yes	device name
devices::id	string	Yes	device id
devices::sub_type	string	Yes	sub type of device
powerchains[]	array	Yes	description of the power chain
powerchains::src-id	string	Yes	source device id
powerchains::src-socket	string	No	num of socket of the source device
powerchains::dst-id	string	Yes	destination device id
powerchains::dst-socket	string	No	num of socket of the source device

Examples :

All examples bellow are illustrated using [dot code](#).

Example 1 : power chain of input infrastructure topology of a DC

Considering a N+1 redundant (Tier II) power input infrastructure topology of a DC :

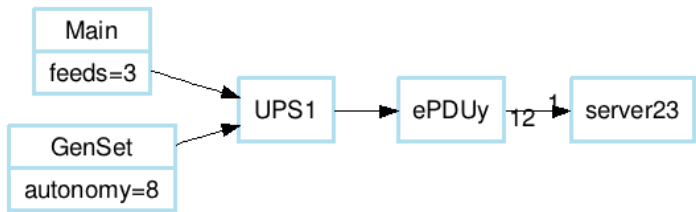
Representation	dot code
<pre> graph LR FEEDB[FEEDB feeds=3] --> STS2[STS2] GenSet2[GenSet2 autonomy=8] --> STS2 STS2 --> UPS2[UPS2] FEEDA[FEEDA feeds=3] --> STS1[STS1] GenSet1[GenSet1 autonomy=8] --> STS1 STS1 --> UPS1[UPS1] </pre>	<pre> digraph Nplus1_TierII { rankdir=LR; //devices definition FEEDA[label = "FEEDA feeds=3" shape = "record"]; FEEDB[label = "FEEDB feeds=3" shape = "record"]; GenSet1[label = "GenSet1 autonomy=8" shape = "record"]; GenSet2[label = "GenSet2 autonomy=8" shape = "record"]; //input power chain definition FEEDA -> STS1; GenSet1 -> STS1; STS1 -> UPS1; FEEDB -> STS2; GenSet2 -> STS2; STS2 -> UPS2; } </pre>

We assume that a "power input" group GR1234 has been created before and each devices involved in the power input have been located in this group.

The request bellow returns the entry input

GET	/api/v1/topology/power?filter_group=GR1234
Response : HTTP 200	
<pre> { "devices": [{ "name": "FEEDA", "id": "MA1234", "sub_type": "feed" }, { "name": "FEEDB", "id": "MA1235", "sub_type": "feed" }, { "name": "GenSet1", "id": "GS1234", "sub_type": "genset" }, { "name": "GenSet2", "id": "GS1235", "sub_type": "genset" }, { "name": "STS1", "id": "ST1234", "sub_type": "sts" }, { "name": "STS2", "id": "ST1235", "sub_type": "sts" }, { "name": "UPS1", "id": "UPS1234", "sub_type": "ups" }, { "name": "UPS2", "id": "UPS1235", "sub_type": "ups" }] "powerchains": [{ "src-id": "MA1234", "dst-id": "ST1234" }, { "src-id": "GS1234", "dst-id": "ST1234" }, { "src-id": "ST1234", "dst-id": "UPS1234" }, { "src-id": "MA1235", "dst-id": "ST1235" }, { "src-id": "GS1235", "dst-id": "ST1235" }, { "src-id": "ST1235", "dst-id": "UPS1235" }] } </pre>	

Example 2 : power chain of a server composed of one Feed and one ePDU.

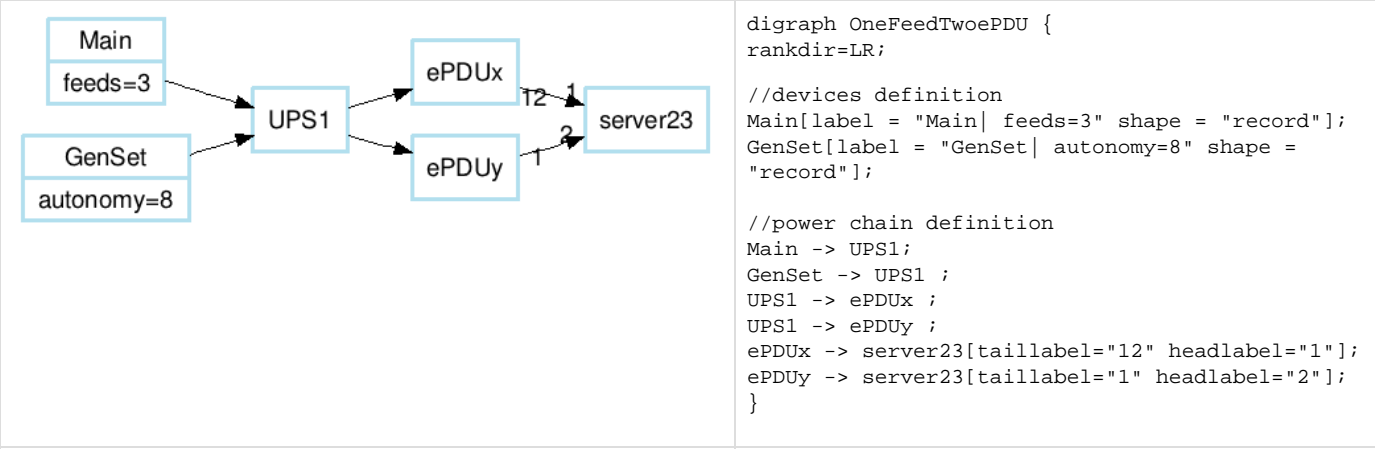
Representation	dot code
 <pre> graph LR Main["Main feeds=3"] --> UPS1["UPS1"] GenSet["GenSet autonomy=8"] --> UPS1 UPS1 --> ePDUy["ePDUy"] ePDUy -- "12" --> "1" server23["server23"] </pre>	<pre> digraph OneFeedOneePDU { rankdir=LR; //devices definition Main[label = "Main feeds=3" shape = "record"]; GenSet[label = "GenSet autonomy=8" shape = "record"]; //power chain definition Main -> UPS1; GenSet -> UPS1 ; UPS1 -> ePDUy ; ePDUy -> server23[taillabel="12" headlabel="1"]; } </pre>

a server23 power topology request should be :

GET	/api/v1/topology/power?to=SRV1234
Response : HTTP 200	
<pre> { "devices": [{ "name": "Main", "id": "MA1234", "sub_type": "feed" }, { "name": "GenSet", "id": "GS1234", "sub_type": "genset" }, { "name": "UPS1", "id": "UPS1234", "sub_type": "ups" }, { "name": "ePDUy", "id": "PDU1234", "sub_type": "pdu" }, { "name": "server23", "id": "SRV1234", "sub_type": "server" }], "powerchains": [{ "src-id": "MA1234", "dst-id": "UPS1234" }, { "src-id": "GS1234", "dst-id": "UPS1234" }, { "src-id": "UPS1234", "dst-id": "PDU1234" }, { "src-id": "PDU1234", "src-socket": 12, "dst-id": "SRV1234", "dst-socket": 1 }] } </pre>	

Example 3 : power chain of a server composed of one Feed and two ePDUs.

Representation	dot code
----------------	----------

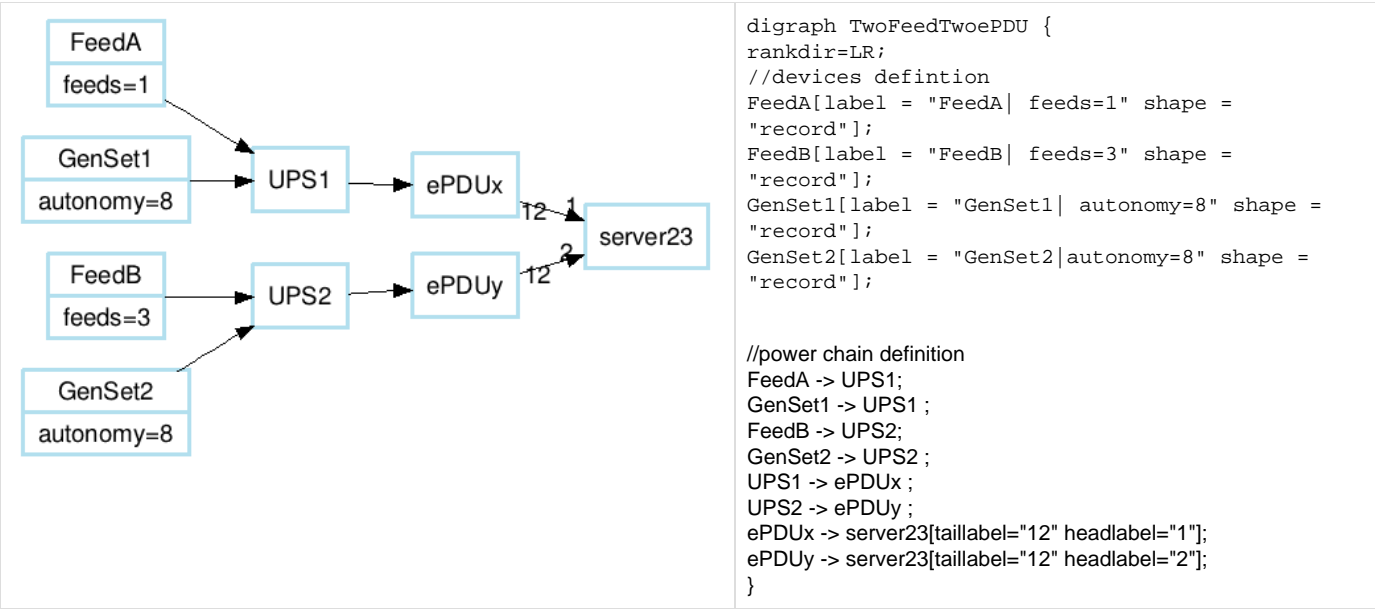


a server23 power topology request should be :

GET	/api/v1/topology/power?to=SRV1234
Response : HTTP 200	
<pre>{ "devices": [{ "name": "Main", "id": "MA1234", "sub_type": "feed" }, { "name": "GenSet", "id": "GS1234", "sub_type": "genset" }, { "name": "UPS1", "id": "UPS1234", "sub_type": "ups" }, { "name": "ePDUx", "id": "PDU1234", "sub_type": "pdu" }, { "name": "ePDUy", "id": "PDU1235", "sub_type": "pdu" }, { "name": "server23", "id": "SRV1234", "sub_type": "server" }], "powerchains": [{ "src-id": "MA1234", "dst-id": "UPS1234" }, { "src-id": "GS1234", "dst-id": "UPS1234" }, { "src-id": "UPS1234", "dst-id": "PDU1234" }, { "src-id": "UPS1234", "dst-id": "PDU1235" }, { "src-id": "PDU1234", "src-socket": 12, "dst-id": "SRV1234", "dst-socket": 1 }, { "src-id": "PDU1235", "src-socket": 1, "dst-id": "SRV1234", "dst-socket": 2 }] }</pre>	

Example 4 : power chain of a server composed of two Feeds and two ePDUs

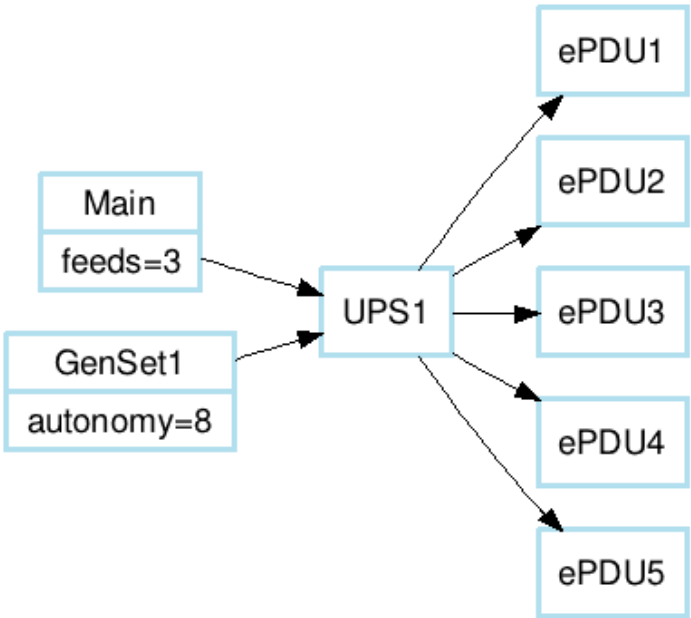
Representation	dot code
----------------	----------



a server23 power topology request should be :

GET	/api/v1/topology/power?to=SRV1234
Response : HTTP 200	
<pre>{ "devices": [{ "name": "FeedA", "id": "MA1234", "sub_type": "feed" }, { "name": "FeedB", "id": "MA1235", "sub_type": "feed" }, { "name": "GenSet1", "id": "GS1234", "sub_type": "genset" }, { "name": "GenSet2", "id": "GS1235", "sub_type": "genset" }, { "name": "UPS1", "id": "UPS1234", "sub_type": "ups" }, { "name": "UPS2", "id": "UPS1235", "sub_type": "ups" }, { "name": "ePDUx", "id": "PDU1234", "sub_type": "pdu" }, { "name": "ePDUy", "id": "PDU1235", "sub_type": "pdu" }, { "name": "server23", "id": "SRV1234", "sub_type": "server" }], "powerchains": [{ "src-id": "MA1234", "dst-id": "UPS1234" }, { "src-id": "GS1234", "dst-id": "UPS1234" }, { "src-id": "MA1235", "dst-id": "UPS1235" }, { "src-id": "GS1235", "dst-id": "UPS1235" }, { "src-id": "GS1234", "dst-id": "UPS1234" }, { "src-id": "UPS1234", "dst-id": "PDU1234" }, { "src-id": "UPS1235", "dst-id": "PDU1235" }, { "src-id": "PDU1234", "src-socket": 12, "dst-id": "SRV1234", "dst-socket": 1 }, { "src-id": "PDU1235", "src-socket": 12, "dst-id": "SRV1234", "dst-socket": 2 }] }</pre>	

Example 4 : power chain of a UPS composed of one Feed, one Genset and five ePDUs

Representation	dot code
	<pre>digraph PowerChainUPS1 { rankdir=LR //device definition Main[label = "Main feeds=3" shape = "record"]; GenSet1[label = "GenSet1 autonomy=8" shape = "record"]; //power chain definition Main -> UPS1 ; GenSet1 -> UPS1 ; UPS1 -> ePDU1; UPS1 -> ePDU2; UPS1 -> ePDU3; UPS1 -> ePDU4; UPS1 -> ePDU5; }</pre>

For this example, two requests are necessary to get the requested power chain.

First, gets the full power chain of UPS1 :

GET	/api/v1/topology/power?to=UPS1234
Response : HTTP 200	
<pre>{ "devices": [{ "name": "Main", "id": "MA1234", "sub_type": "feed" }, { "name": "GenSet1", "id": "GS1234", "sub_type": "genset" }, { "name": "UPS1", "id": "UPS1234", "sub_type": "ups" }], "powerchains": [{ "src-id": "MA1234", "dst-id": "UPS1234" }, { "src-id": "GS1234", "dst-id": "UPS1234" }] }</pre>	

Second, gets list of devices power by UPS1 :

GET	/api/v1/topology/power?from=UPS1234
Response : HTTP 200	

```
{
  "devices": [
    { "name": "UPS1", "id": "UPS1234", "sub_type": "ups" },
    { "name": "ePDU1", "id": "PDU1234", "sub_type": "pdu" },
    { "name": "ePDU2", "id": "PDU1235", "sub_type": "pdu" },
    { "name": "ePDU3", "id": "PDU1236", "sub_type": "pdu" },
    { "name": "ePDU4", "id": "PDU1237", "sub_type": "pdu" },
    { "name": "ePDU5", "id": "PDU1238", "sub_type": "pdu" } ],
  "powerchains": [
    { "src-id": "UPS1234", "dst-id": "PDU1234" },
    { "src-id": "UPS1234", "dst-id": "PDU1235" },
    { "src-id": "UPS1234", "dst-id": "PDU1236" },
    { "src-id": "UPS1234", "dst-id": "PDU1237" },
    { "src-id": "UPS1234", "dst-id": "UPS1238" } ]
}
```

Example 5 : a failed topology request on an unknown device ID.

GET	/api/v1/topology/power?from=<unknown_ID> /api/v1/topology/power?to=<unknown_ID>
Response : HTTP 404, not found	

Example 6 : a failed topology request on a non device element, here GR1234 is a group.

GET	/api/v1/topology/power?from=GR1234 /api/v1/topology/power?to=GR1234
Response : HTTP 400, bad input	

input_power_chain (Read)

Returns input power chain for datacenter with ID dc_id.

There two ways of building the power chain response:

- input power group is defined in csv devices within this group will be used in the chain
- no input power group defined all devices which are right behind power source (feed, genset, sts ...) will be used in the chain of datacenter

These two approaches mentioned above cannot be combined, that means if csv with input power group is imported and then new device is added manually from UI, it will not be included into the input power chain.

If the input power group is empty the input power chain will be empty too.

Operation	Method	URI	Description
Read	GET	/api/v1/topology/input_power_chain/<dc_id>	Get input power chain for datacenter with ID dc_id.

Request parameter:

- <dc_id> - ID of datacenter

Example:

GET	/api/v1/topology/input_power_chain/3
Response:	HTTP 200
	<pre>{ "devices" : [{ "name" : "MAIN01-LAB", "id" : "10", "sub_type" : "feed"}, { "name" : "GENSET-LAB", "id" : "11", "sub_type" : "genset"}, { "name" : "BladeUPS", "id" : "12", "sub_type" : "ups"}, { "name" : "INPUT-UPS", "id" : "13", "sub_type" : "ups"}] }</pre>

Metric(v1)
current(Read)

Returns the actual value of all available metrics for specified element

Operation	Method	URI	Description
Read	GET	/api/v1/metric/current?dev={dev-id}&src-id={src-id}&dst-id={dst-id}&dst-socket={dst-socket}	Retrieves current values of metrics for requested element

Version : 1

Security : Public access (level 0)

Request Parameters :

- dev: list of asset ids.

Response Document : JSON doc.

For a UPS, the response document contains :

Property name	type	mandatory	description
id	String	Yes	element id
name	string(50)	Yes	element name
status.ups	string	Yes	UPS status (to be described) OL - On line (mains is present) OB - On battery (mains is not present) LB - Low battery HB - High battery RB - The battery needs to be replaced CHRG - The battery is charging DISCHRG - The battery is discharging (inverter is providing load power) BYPASS - UPS bypass circuit is active - no battery protection is available CAL - UPS is currently performing runtime calibration (on battery) OFF - UPS is offline and is not supplying power to the load OVER - UPS is overloaded TRIM - UPS is trimming incoming voltage (called "buck" in some hardware) BOOST - UPS is boosting incoming voltage FSD - Forced Shutdown (restricted use, see the note below)
temperature.default	number	No	UPS internal temperature (degrees C)
load.default	number	Yes	Load on UPS (percent)
realpower.default	number	Yes	Total Real power (Watts) output. Total of (realpower.output.Lx)
voltage.output.L1-N	number	Yes	Voltage (V) of the output feed L1 with neutral (the default one in single phase)
realpower.output.L1	number	Yes	Real power (W) of the output feed L1 (the default one in single phase)
current.output.L1	number	Yes	Current (A) of output feed L1 (the default one in single phase)
voltage.output.L2-N	number	No	Voltage (V) of the output feed L2 with neutral
realpower.output.L2	number	No	Real power (W) of the output feed L2
current.output.L2	number	No	Current (A) of output feed L2
voltage.output.L3-N	number	No	Voltage (V) of the output feed L3 with neutral

realpower.output.L3	number	No	Real power (W) of the output feed L3
current.output.L3	number	No	Current (A) of output feed L3
charge.battery	number	Yes	Battery charge (percent)
runtime.battery	number	Yes	Battery remaining runtime duration (second)

For an ePDU the response document contains :

Property name	type	mandatory	description
id	String	Yes	device id
name	string(50)	Yes	device name
frequency.input	number	Yes	Frequency (Hz) of the input
load.input.L1	number	Yes	Load (%) of input feed L1 (the default one in single phase)
load.input.L2	number	No	Load (%) of input feed L2
load.input.L3	number	No	Load (%) of input feed L3
voltage.input.L1-N	number	Yes	Voltage (V) of the input feed L1 with neutral (the default one in single phase)
voltage.input.L2-N	number	No	Voltage (V) of the input feed L2 with neutral
voltage.input.L3-N	number	No	Voltage (V) of the input feed L3 with neutral
current.input.L1	number	Yes	Current (A) of input feed L1 (the default one in single phase)
current.input.L2	number	No	Current (A) of input feed L2
current.input.L3	number	No	Current (A) of input feed L3
realpower.default	number	Yes	Real power (W) of the input (the total of each phase)
realpower.input.L1	number	Yes	Real power (W) of the input feed L1 (the default one in single phase)
realpower.input.L2	number	No	Real power (W) of the input feed L2
realpower.input.L3	number	No	Real power (W) of the input feed L3
power.default	number	Yes	Power (W) of the input (the total of each phase)
power.input.L1	number	Yes	Power (W) of the input feed L1 (the default one in single phase)
power.input.L2	number	No	Power (W) of the input feed L2
power.input.L3	number	No	Power (W) of the input feed L3
outlets	object	Yes	Providing information about outlets. Key is number of outlet, value is Outlet properties object described below.

Outlet properties

Property name	type	mandatory	description
realpower	number	No	Real power (W) of the outlet X - null if not available
current	number	No	Current (A) of the outlet - null if not available
voltage	number	No	Voltage (V) of the outlet - null if not available
status	string	No	Status of the outlet ("on", "off") - null if not available

For a Datacenter the response document contains :

Property name	type	mandatory	description
id	String	Yes	asset id
name	string(50)	Yes	asset name
realpower.default	number	Yes	Real power (W) of the output (the total of each phase)

realpower.output.L1	number	Yes	Real power (W) of the output feed L1 (the default one in single phase)
realpower.output.L2	number	No	Real power (W) of the output feed L2 (only for 3 phase)
realpower.output.L3	number	No	Real power (W) of the output feed L3 (only for 3 phase)

Example 1 : get current value of Eaton 3S 550 UPS and ePDU

GET	/api/v1/metric/current?dev=UPS1234,PDU1234
Response : HTTP 200	
<pre>{ "current": [{ "id": "UPS1234", "name": "ups1", "status.ups": "OL CHRG", "load.default": 0, "voltage.output": 115.0, "charge.battery": 100 }, { "id": "PDU1234", "name": "pdul", "realpower.outlet": 100 }] }</pre>	

Condition	HTTP code	code	message
No argument provided	400/Bad Request	46	Parameter '<list-of-ids>' is required.

All other errors like not numeric id, unknown id, ... are not reported and such id is simply skipped from result document.

computed

All computed value get the same generic approach, the URI for calculated values has this form:

GET	/api/v1/metric/computed/<value_name>?arg1=A[,B[, ...]] [&arg2=D[,E[, ...]]] ... [&argN=A[,B[, ...]]]
------------	---

rack_total(Read)

For total rack power we have the value called "rack_total"

parameter	type	mandatory	description	example
arg1	<dev_id>,<dev_id>	Yes	This is list of rack ID separated by ","	arg1=4 arg1=4,6,1
arg2	<value1>,<value2>	Yes	Requested values. This is list of values separated by "," see next table for detail	arg2=total_power,avg_last_day

requested value	meaning	Suggested names
-----------------	---------	-----------------

total_power	Actual power consumption (W)	
avg_power_last_day	average power consumption for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (W)	avg_power_last_24h
avg_power_last_week	average power consumption for last 7 days in the interval [NOW -7days,NOW] (including both ends) (W)	avg_power_last_7d
avg_power_last_month	average power consumption for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (W)	avg_power_last_30d
avg_power_last_year	average power consumption for last 365 days in the interval [NOW-365Days,NOW] (including both ends) (W)	avg_power_last_365d

Example 1 : request rack ID RC1234 for all values

GET	/api/v1/metric/computed/rack_total?arg1=RC1234 &arg2=total_power,avg_power_last_day,avg_power_last_week,avg_power_last_year,avg_power_last_month
Response : HTTP 200	
<pre>{ "rack_total": [{ "id": "RC1234", "name": "my rack 1234", "total_power": 39.3, "avg_power_last_day": 33.5, "avg_power_last_week": 38.6, "avg_power_last_month": 39.2, "avg_power_last_year": 34.9 }] }</pre>	

Example 2 : request racks ID RC2 and RC4 for total_power

GET	/api/v1/metric/computed/rack_total?arg1=RC2,RC4&arg2=total_power
Response : HTTP 200	
<pre>{ "rack_total": [{ "id": "RC2", "name": "my rack 2", "total_power": 39.3 }, { "id": "RC4", "name": "my rack 4", "total_power": 31.6 }] }</pre>	

datacenter_indicators (Read)

For datacenter we have the value called "datacenter_indicators"

parameter	type	mandatory	description	example
-----------	------	-----------	-------------	---------

arg1	<dc_id>,<dc_id>	Yes	This is list of datacenter ID separated by ","	arg1=4 arg1=4,6,1
arg2	<value1>,<value2>	Yes	Requested values. This is list of values separated by "," see next table for detail	arg2=total_power,avg_last_day

requested value	meaning
power	Actual DC power consumption (W)
avg_power_last_day	average power consumption for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (W)
avg_power_last_week	average power consumption for last 7 days in the interval [NOW -7days,NOW] (including both ends) (W)
avg_power_last_month	average power consumption for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (W)
min_power_last_day	minimal power consumption for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (W)
min_power_last_week	minimal power consumption for last 7 days in the interval [NOW -7days,NOW] (including both ends) (W)
min_power_last_month	minimal power consumption for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (W)
max_power_last_day	maximal power consumption for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (W)
max_power_last_week	maximal power consumption for last 7 days in the interval [NOW -7days,NOW] (including both ends) (W)
max_power_last_month	maximal power consumption for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (W)
trend_power_last_day	power consumption trend for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (W)
trend_power_last_week	power consumption trend for last 7 days in the interval [NOW -7days,NOW] (including both ends) (W)
trend_power_last_month	power consumption trend for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (W)
temperature	average actual temperature of the DC (°C)
avg_temperature_last_day	average temperature for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (°C)
avg_temperature_last_week	average temperature for last 7 days in the interval [NOW -7days,NOW] (including both ends) (°C)
avg_temperature_last_month	average temperature for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (°C)
min_temperature_last_day	minimal temperature for last 24 hours in the interval [NOW – 24h, NOW] (including both ends)
min_temperature_last_week	minimal temperature for last 7 days in the interval [NOW -7days,NOW] (including both ends) (°C)
min_temperature_last_month	minimal temperature for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (°C)
max_temperature_last_day	maximal temperature for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (°C)
max_temperature_last_week	maximal temperature for last 7 days in the interval [NOW -7days,NOW] (including both ends) (°C)
max_temperature_last_month	maximal temperature for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (°C)
trend_temperature_last_day	temperature trend for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (°C)
trend_temperature_last_week	temperature trend for last 7 days in the interval [NOW -7days,NOW] (including both ends) (°C)
trend_temperature_last_month	temperature trend for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (°C)
humidity	average actual humidity of the DC (%)
avg_humidity_last_day	average humidity for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (°C)
avg_humidity_last_week	average humidity for last 7 days in the interval [NOW -7days,NOW] (including both ends) (°C)
avg_humidity_last_month	average humidity for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (°C)
min_humidity_last_day	minimal humidity for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (°C)
min_humidity_last_week	minimal humidity for last 7 days in the interval [NOW -7days,NOW] (including both ends) (°C)
min_humidity_last_month	minimal humidity for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (°C)

max_humidity_last_day	maximal humidity for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (°C)
max_humidity_last_week	maximal humidity for last 7 days in the interval [NOW -7days,NOW] (including both ends) (°C)
max_humidity_last_month	maximal humidity for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (°C)
trend_humidity_last_day	humidity trend for last 24 hours in the interval [NOW – 24h, NOW] (including both ends) (°C)
trend_humidity_last_week	humidity trend for last 7 days in the interval [NOW -7days,NOW] (including both ends) (°C)
trend_humidity_last_month	humidity trend for last 30 days in the interval [NOW-30Days,NOW] (including both ends) (°C)

Example 1 : request DC ID 19 for current total power

GET	/api/v1/metric/computed/datacenter_indicators?arg1=19&arg2=power
Response : HTTP 200	
<pre>{ "datacenter_indicators": [{ "id": "19", "name": "DC-LAB", "power": 3445.6 }] }</pre>	

average (Read) - Work in Progress

Operation	Method	URI	Description
Read	GET	/api/v1/metric/computed/average	Read all stored averaged data from within the specified interval, of the specified type and specified granularity.

Version : 1

Security : Public access (level 0)

Request parameters:

key	description
start_ts	Start of the interval. Format as per RFC-11 standard.
end_ts	End of the interval. Format as per RFC-11 standard.
type	Type of requested average. One of the following values: "arithmetic_mean" "min" "max"
step	Granularity of the requested average. One of the following values: "15m" "30m" "1h" "8h" "24h" <ul style="list-style-type: none"> 15m - fifteen minutes 30m - thirty minutes 1h - one hours 8h - eight hours 24h - twentyfour hours
element_id	ID of the requested element
source	requested source

relative	<p>One of the following values: "24h" "7d" "30d" "" (empty string).</p> <p>Value "" (empty string) behaves as if the parameter 'relative' was not specified at all.</p> <p>Overrides parameters 'start_ts', 'end_ts'. Sets parameter 'end_ts' to current timestamp (i.e. now), parameter 'start_ts' to current timestamp minus the value of 'relative' parameter expressed in seconds.</p> <p>You do not have to specify parameters 'start_ts', 'end_ts' when specifying parameter 'relative', but specifying them does not create conflict.</p>
ordered	<p>It is optional parameter. Available values: "true"/"false". Default value = 'false'. If 'ordered=true' then returned data are ordered by timestamp in ascending order.</p>

Constraints:

- end_ts must be always greater than start_ts
- (*) Basically ISO 8601 without slash (-) characters and with hardcoded UTC timezone appended as suffix 'Z'. Example: "20150101121314Z". Just a reminder: 000000 (as in hhmmss) is always start of a day and 240000 is always end of a day. Therefore the following equivalence holds true "20150101240000Z" <==> "20150102000000Z".

Response Document : JSON document

Root type must be Object.

All key / values mandatory.

key	type	description
element_id	string	Requested average type. This value is returned even if the optional request parameter is omitted. At the moment this property can hold only one possible value "arithmetic_mean".
start_ts	string	Beginning of the interval we are interested in.
end_ts	string	End of the interval we are interested in.
units	string	Units in which is the measurement done - ex C
type	string	Type of the computed values - ex. arithmetic_mean
step	string	Granularity of the averages - ex. 8h
source	string	Type of data requested - ex. temperature.thermal_zone0. Source must match the regular expression "(^realpower.default.* .temperature.* .humidity.*)"
data	array	
data::value	number	Value measured/computed == data::value * 10^data::scale
data::scale	number	Value measured/computed == data::value * 10^data::scale
data::timestamp	number	Unixtime of when are data valid

Example:

command

```
curl
'http://127.0.0.1:8000/api/v1/metric/computed/average?end_ts=2016010100000
0Z&start_ts=20130101000000Z&type=arithmetic_mean&step=8h&element_id=26&so
urce=temperature.thermal_zone'
```

JSON output

```
{
  "units": "C",
  "source": "temperature.thermal_zone",
  "step": "8h",
  "type": "arithmetic_mean",
  "element_id": 26,
  "start_ts": 1356991200,
  "end_ts": 1451602800,
  "data": [
    {
      "value": 560,
      "scale": -1,
      "timestamp": 1426201200
    },
    {
      "value": 480,
      "scale": -1,
      "timestamp": 1426201680
    },
    {
      "value": 510,
      "scale": -1,
      "timestamp": 1426202160
    },
    {
      "value": 650,
      "scale": -1,
      "timestamp": 1426287600
    }
  ]
}
```

Assumption: Command `date -u +%Y%m%d%H%M%SZ` produces "20160106131438Z"

command

```
curl
'http://127.0.0.1:8000/api/v1/metric/computed/average?type=arithmetic_mean
&step=24h&element_id=26&source=temperature.thermal_zone&relative=7d'
```

JSON output

```
{
  "units": "C",
  "source": "temperature.thermal_zone",
  "step": "24h",
  "type": "arithmetic_mean",
  "element_id": 26,
  "start_ts": 1451481278,
  "end_ts": 1452086078,
  "data": [
    {
      "value": 7825,
      "scale": -2,
      "timestamp": 1451520000
    },
    {
      "value": 670,
      "scale": -1,
      "timestamp": 1451606400
    },
    {
      "value": 510,
      "scale": -1,
      "timestamp": 1451692800
    },
    {
      "value": 340,
      "scale": -1,
      "timestamp": 1451779200
    },
    {
      "value": 2714,
      "scale": -2,
      "timestamp": 1451865600
    },
    {
      "value": 324,
      "scale": -1,
      "timestamp": 1451952000
    },
    {
      "value": 732,
      "scale": -1,
      "timestamp": 1452038400
    }
  ]
}
```

Response

Error: [Error message template](#)

Condition	HTTP code	code	message
Value of 'start_ts' not valid.	400	47	Parameter 'start_ts' has bad value. Received value '<start_ts>'. Expected '^([0-9]{14}Z))\$'.
Value of 'end_ts' not valid.	400	47	Parameter 'start_ts' has bad value. Received value '<end_ts>'. Expected '^([0-9]{14}Z))\$'.
Value of 'type' not valid.	400	47	Parameter 'start_ts' has bad value. Received value '<type>'. Expected '^(arithmetic_mean min max))\$'.
Value of 'step' not valid.	400	47	Parameter 'step' has bad value. Received value '<step>'. Expected '^([0-9]{1,2}[a-z]))\$'.
Value of 'source' not valid.	400	47	Parameter 'source' has bad value. Received value '<step>'. Expected '^[_@a-z0-9]{0,255}\$'.
Value of 'relative' not valid.	400	47	Parameter 'source' has bad value. Received value '<step>'. Expected '^([0-9]{1,2}[a-z]))\$'.
Value of 'element_id' not valid.	400	47	Parameter 'element_id' has bad value. Received value '<element_id>'. Expected valid id.
Value of 'ordered' not valid.	400	47	Parameter 'ordered' has bad value. Received value '<ordered>'. Expected '(true false)'.
Requested source not present.	404	54	Data for type '<type>', ... element_id='<element_id>'.

If source is already in backend database, this call returns HTTP 200 OK with empty data array.

uptime(Read)

time for outage and for total are returned in **seconds**

parameter	type	mandatory	description	example
arg1	<dc_id>,<dc_id>	Yes	This is list of rack ID separated by ","	arg1=4 arg1=4,6,1

Example 1 : Request data for DC with id = "DC1"

GET	/api/v1/metric/computed/uptime?arg1=DC1
Response : HTTP 200	
<pre>{ "outage": [{ "id": "DC1", "name": "My favourite DC", "outage": 4000, "total": 1123456 }] }</pre>	

Errors :

TODO

Alert (v1)

activelist (Read)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/alerts/activelist	Retrieve alerts list. Optionally limit to certain state, element or all elements belonging under specified element in the topology tree.	level 2

Request parameters

- **state** (optional) - possible values 'ALL', 'ALL-ACTIVE', 'ACTIVE', 'ACK-WIP', 'ACK-IGNORE', 'ACK-PAUSE', 'ACK-SILENCE', 'RESOLVED'. Empty value equals to value 'ALL-ACTIVE'.
- **asset** (optional) - element identifier. List alerts for this asset element.
- **recursive** (optional) - List alerts recursively for all asset elements that belong under this asset element in topology. Possible values: 'true', 'false'. Default value: 'false'.

Note: State 'RESOLVED' can not be set manually - it is a result of conditions, that triggered an alert, coming back to normal values.

Response

Error: [Error message template](#)

Condition	HTTP code	code	message
Value of 'state' not valid.	400	47	Parameter 'state' has bad value. Received value '<state>'. Expected one of the following values [ALL ALL-ACTIVE ACTIVE ACK-WIP ACK-IGNORE ACK-PAUSE ACK-SILENCE RESOLVED].
Value of 'recursive' not valid.	400	47	Parameter 'recursive' has bad value. Received value '<recursive>'. Expected one of the following values [true false].
Value of 'asset' is not a valid element identifier.	400	47	Parameter 'asset' has bad value. Received value is not an element identifier. Expected an unsigned integer in range 1 to UINT_MAX.
Value of 'asset' is out of range.	400	47	Parameter 'asset' has bad value. Received value is out of range. Expected value in range 1 to UINT_MAX.

Success : JSON doc

root:

- type: Array
- items: Objects

Property name	type	description
timestamp	string	date + time when alert was detected in the system
rule_name	string	Name of the rule which generates the alert
element_id	string	element id from where comes the alert. Empty string means a pure software alert
element_name	string	element name. Empty string means a pure software alert
element_type	string	provides element type : datacenter, room, group, row, rack, device. Empty string means a pure software alert
element_sub_type	string	provides the kind of element for group and device element. Empty string means a pure software alert.
severity	string	CRITICAL , WARNING
description	string	description
state	string	ACTIVE, ACK-WIP, ACK-SILENCE, ACK-PAUSE, ACK-IGNORE, RESOLVED

Example :

GET	/api/v1/alerts/activelist?asset=1&recursive=true
Response : HTTP 200	

```
[
  {
    "timestamp": "1970-01-01T00:00:01Z",
    "rule_name": "Threshold",
    "element_id": "7",
    "element_name": "ups",
    "element_type": "device",
    "element_sub_type": "ups",
    "state": "ACTIVE",
    "severity": "high",
    "description": "description"
  },
  {
    "timestamp": "1970-01-01T00:00:02Z",
    "rule_name": "Pattern",
    "element_id": "6",
    "element_name": "epdu",
    "element_type": "device",
    "element_sub_type": "epdu",
    "state": "ACTIVE",
    "severity": "high",
    "description": "description"
  },
  {
    "timestamp": "1970-01-01T00:00:03Z",
    "rule_name": "Single",
    "element_id": "7",
    "element_name": "ups",
    "element_type": "device",
    "element_sub_type": "ups",
    "state": "ACTIVE",
    "severity": "high",
    "description": "description"
  }
]
```

rules (Read, Create, Update)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/alerts/rules[?type=<type>][&rule_class=<class>]	List rule representations, optionally limited to certain type or class	level 2

Request parameters

- **type** (optional) - possible values 'threshold', 'single', 'pattern', 'all'. Empty value equals to value 'all'.
- **rule_class** (optional) - list the rules with defined class only. No class means give me all. The class is opaque string and equals to rule_class in the rule itself.

Response

Error: [Error message template](#)

Condition	code	message
Unknown <type> value	47	Parameter 'type' has bad value. Received type '%s'. Expected one of the following values ['threshold', 'single', 'pattern', 'all'].

Success : JSON doc

root:

- type: Array

Example :

GET	/api/v1/alerts/rules
Response : HTTP 200	
<pre>[{ "threshold" : { } }, { "threshold" : { } }, { "pattern" : { } }, { "single" : { } }]</pre>	


```

["<action_1>", ..., "<action_N>"], "description" : "<description>" }},
      { "high_critical" : { "action" :
["<action_1>", ..., "<action_N>"], "description" : "<description>" }} ],
      "evaluation"      :   "<lua_function>",
      "rule_source"     :   "<rule_source>"
    }
  }
{
  "pattern" : {
    "rule_name"      :   "<rule_name>",
    "target"         :   "<regex>",
    "values"         :   [ { "<value_name_1>" : "<value>" },
                          ...,
                          { "<value_name_N>" : "<value>" } ],
    "results"        :   [ { "<result_name_1>" : { "action" :
["<action_1>", ..., "<action_N>"], "description" : "<description>" }},
                          ...,
                          { "<result_name_N>" : { "action" :
["<action_1>", ..., "<action_N>"], "description" : "<description>" }} ],
    "evaluation"      :   "<lua_function>",
    "rule_source"     :   "<rule_source>"
  }
}
{
  "single" : {
    "rule_name"      :   "<rule_name>",
    "target"         :   [ "<topic_1>", ... , "<topic_N>" ],
    "values"         :   [ { "<value_name_1>" : "<value>" },
                          ...,
                          { "<value_name_N>" : "<value>" } ],
    "element"        :   "<element_name>",
    "results"        :   [ { "<result_name_1>" : { "action" :
["<action_1>", ..., "<action_N>"], "description" : "<description>" }},
                          ...,
                          { "<result_name_N>" : { "action" :
["<action_1>", ..., "<action_N>"], "description" : "<description>" }} ],
    "evaluation"      :   "<lua_function>",
    "rule_source"     :   "<rule_source>"
  }
}

```

where

* <rule_name>	unique rule name, CASE INSENSITIVE
* <topic_specification>	string value representing topic OR array of topics
* <element_name>	name of element
* <value>	number (integer, floating-point)
* <action_X>	currently EMAIL and SMS are defined. Determines the resulting alert action. Other (3d party) types are allowed.
* <lua_function>	lua function that evaluates trigger conditions
* <topic_X>	metric topic
* <regex>	regular expression
* <value_name_X>	unique name (identifier) of value within given rule definition. Case sensitive.

* <result_name_X> one of the following ["low_critical", "low_warning", "high_critical", "high_warning"]. Case sensitive.

* <rule_source> where this rule comes from: "Manual user input"/"NUT"/ (can be added later). The "Manual user input" rules can be edited manually, "NUT" rules are not allowed to be changed by user.

For threshold rule:

If <topic_specification> is single string value then property "evaluation" : "<lua_function>" is ignored.

If <topic_specification> is array then property "evaluation" : "<lua_function>" is mandatory. For further details please consult the section describing semantics.

The semantics is as follows:

GENERAL:

For result "high_critical", "low_critical" severity of generated alert is "CRITICAL".

For result "high_warning", "low_warning" severity of generated alert is "WARNING".

THRESHOLD:

* When <topic_specification> is one string:

If value of metric specified by <topic_specification> is higher than value of "high_warning", "high_critical" values or lower than value of "low_warning", "low_critical" values, then alert is triggered for element <element_name> with state ACTIVE and description and action equal to those specified in "results" item "high_warning" , "high_critical" or "low_warning", "low_critical" respectively. When value of metric specified by <topic_specification> returns back

to normal (i.e higher than value of "low_warning" and lower than value of "high_warning") trigger the same alert with state RESOLVED.

* When <topic_specification> is an array of strings:

If result of <lua_function> is one of LOW_CRITICAL, LOW_WARNING, HIGH_CRITICAL, HIGH_WARNING, then alert is triggered for element <element_name> with state ACTIVE and description and action equal to those specified in "results" item "high_warning", "high_critical" or "low_warning", "low_critical" respectively. If subsequently result of <lua_function> is OK then the same alert is triggered with state RESOLVED.

If result of <lua_function> is anything else, the rule is ignored.

Values of properties <value_name_1>,...,<value_name_N> become available in <lua_function> as global variables "<value_name_1>", ..., "<value_name_N>"

PATTERN:

Value of metric, whose topic matches <regex>, becomes available in <lua_function> as first parameter. Values of properties <value_name_1>,..., <value_name_N> become available in <lua_function> as global variables "<value_name_1>", ..., "<value_name_N>". Using these values you MUST return

one of the following constants: LOW_CRITICAL, LOW_WARNING, HIGH_CRITICAL, HIGH_WARNING, OK.

If <lua_function> returns one of LOW_CRITICAL, LOW_WARNING, HIGH_CRITICAL, HIGH_WARNING then alert is triggered for element specified by matched topic with state ACTIVE and description and action equal to those specified in "result" item returned.

If subsequently result of <lua_function> is OK then the same alert is triggered with state RESOLVED.

If result of <lua_function> is anything else, the rule is ignored.

SINGLE:

Analogously to PATTERN rule, only target is not a regular expression but a predefined set of topics.

How to write <lua_function>: It is a lua program with name "main" and arguments. Order of arguments is the same as order of topics in "target" property. And topic Nth from "targets" is assigned to Nth argument to the function "main". Function main should always return a result. Possible

results are LOW_CRITICAL, LOW_WARNING, HIGH_CRITICAL, HIGH_WARNING, OK.
You can use global variables defined in "values".

Operation	Method	URI	Description	Security
Read	GET	/api/v1/alerts/rules/<name>	Representation of rule with name <name>	level 2

Notes: rule name is unique across the system

Response

Error: [Error message template](#)

Condition	HTTP code	code	message
<name> empty	400	47	Parameter 'name' has bad value. Received empty string. Expected non-empty valid rule name. For a list of all available rule names please perform GET alerts/rules.
Rule <name> does not exist	404	54	Rule name 'name' does not exist.

Success: JSON document - [Alerts Rule Format](#)

Example 1: Simple threshold rule

GET	/api/v1/alerts/rules/Threshold_1
Response : HTTP 200	
<pre>{ "threshold" : { "rule_name" : "Threshold_1", "target" : "temperature.default@ups-9", "element" : "ups-9", "values" : [{ "low_critical" : "-20" }, { "low_warning" : "0" }, { "high_warning" : "55" }, { "high_critical" : "90" }], "results" : [{ "low_critical" : { "action" : ["EMAIL", "SMS"], "description" : "blah" }, { "low_warning" : { "action" : ["EMAIL"], "description" : "blah blah" }}, { "high_warning" : { "action" : ["EMAIL", "SMS"], "description" : "chicken" }}, { "high_critical" : { "action" : ["EMAIL", "SMS"], "description" : "FIRE-BRIGADE-DPT-CONSOLE" } }] } }</pre>	

Example 2:Complex threshold rule

GET	/api/v1/alerts/rules/Threshold_2
Response : HTTP 200	

```
{
  "threshold" : {
    "rule_name"      : "Threshold_2",
    "target"         : [ "aaa@ups_in_the_lab", "bbb@ups_in_the_lab", "ccc@ups_in_the_lab",
    "element"        : "ups_in_the_lab",
    "values"         : [ { "high_warning" : "2,25" },
                        { "high_critical" : "2,27" } ],
    "results"        : [ { "high_warning" : { "action" : [ "EMAIL", "SMS" ], "description" :
"Something is about to happen" } },
                        { "high_critical" : { "action" : [ "EMAIL", "SMS",
"FIRE-BRIGADE-DPT-CONSOLE" ], "description" : "It is happened!" } } ],
    "evaluation"     : "function main(abc_fff1,abc_fff2) local new_value = abc_fff1 + abc_fff2 if (
new_value < low_critical ) then return LOW_CRITICAL end if ( new_value < low_warning ) then return
LOW_WARNING end if ( high_critical < new_value ) then return HIGH_CRITICAL end if ( high_warning <
new_value ) then return HIGH_WARNING end return OK end"
  }
}
```

Operation	Method	URI	Description	Security
Create	POST	/api/v1/alerts/rules	Create new rule	level 2

Request Document: JSON document - [Alerts Rule Format](#)

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)
Request document not valid json or does not conform to the specified format/schema.	48	Request document has invalid syntax. Please check RFC-11 for valid rule json schema description.
Request document has valid json, but lua function has syntax errors.	48	Request document has lua syntax error.
Rule with this name already exists in the system	52	Rule with such name (new rule name) already exists"

Success - Document : Identical to [Alerts Rule Format](#)

Operation	Method	URI	Description	Security
Update	PUT	/api/v1/alerts/rules/<name>	Update existing rule, provided <name> exists.	level 2

Request Document: JSON document - [Alerts Rule Format](#)

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)
Request document not valid json or does not conform to the specified format/schema.	48	Request document has invalid syntax. Please check RFC-11 for valid rule json schema description.
Name 'name' does not exist.	54	Rule name 'name' does not exist.
Request document has valid json, but lua function has syntax errors.	48	Request document has lua syntax error.
Rule with new rule name already exists in the system	52	Rule with such name (new rule name) already exists

Success - Document : Identical to [Alerts Rule Format](#)

acknowledge (Update)

Operation	Method	URI	Description	Security
-----------	--------	-----	-------------	----------

Update	PUT	/api/v1/alerts/ack/<rule_name>/<element_name>	Set acknowledge state to alert identified by <rule_name> and <element_name>	level 2
--------	-----	---	---	---------

Request Document: JSON document

```
{
  "state" : "<state>"
}
```

where <state> must be one of [ACTIVE | ACK-WIP | ACK-IGNORE | ACK-PAUSE | ACK-SILENCE].

Response

Error: [Error message template](#)

Condition	HTTP code	code	message
Bad request document	HTTP_BAD_REQUEST	48	Request document has invalid syntax. Please check RFC-11 for valid json schema description.
Value of <state> is not valid	HTTP_BAD_REQUEST	47	Parameter 'state' has bad value. Received <bad_value>. Expected one of the following values [ACTIVE ACK-WIP ACK-IGNORE ACK-PAUSE ACK-SILENCE].
Alert identified by <rule_name> and <element_name> does not exist.	HTTP_NOT_FOUND	54	Alert identified by rule name = '<rule_name>', element name = '<element_name>' does not exist.
Alert identified by <rule_name> and <element_name> can not change state to <state>.	HTTP_BAD_REQUEST	52	Alert identified by rule name = '<rule_name>', element name = '<element_name>' can not change state to <state>.

Success : JSON doc

root:

- type: Object
- properties: "rule_name", "element_name", "state"

property	type	Value
rule_name	String	name of the rule identifying alert
element_name	String	name of the element identifying alert
state	String	state of the alert

Admin (v1)

time (Read, Update)

Operation	Method	URI	Description
Read	GET	/api/v1/admin/time	Retrieves RC internal time

Version : 1

Security : Public access (level 0)

Response Document : JSON doc

Property name	type	description
---------------	------	-------------

time	timestamp	current date time
------	-----------	-------------------

Example :

GET	/api/v1/admin/time
Response : HTTP 200	
<pre>{ "time": "2014-10-20T12:30:00Z", "ntp": "pool.ntp.org" }</pre>	

Operation	Method	URI	Description
Set up	POST	/api/v1/admin/time	Set up an date internal system time

Version : 1

Security : restricted access, with minimum security of level 2

Request Document : JSON doc

root:

- type: Object
- properties: "time", "ntp";

Property name	type	mandatory	description
time	String	One of	date time; required format: <YYYY>-<MM>-<DD>T<hh>:<mm>:<ss>Z
ntp	String	One of	ntp server (ip address or hostname)

Example 1 : set a UTC timestamp

PUT	/api/v1/admin/time
<pre>{ "time": "2014-10-20T12:30:00Z" }</pre>	
Response : HTTP 200	
<pre>{ "time": "2014-10-20T12:30:00Z", "ntp": "pool.ntp.org" }</pre>	

Example 2 : set a NTP server

PUT	/api/v1/admin/time
<pre>{ "ntp": "pool.ntp.org" }</pre>	
Response : HTTP 200	
<pre>{ "time": "2014-10-20T12:30:00Z", "ntp": "pool.ntp.org" }</pre>	

config (Read, Update)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/admin/config?key=<key>	Retrieves value of specified configuration setting	level 2
Update	POST	/api/v1/admin/config	Updates values of specified configuration settings or creates new configuration settings	level 2

Read

Request Parameters:

- **key**: requested configuration setting

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)	Corresponding HTTP status code
<key> empty or setting does not exist	TODO	TODO	404

Success - Document : JSON doc

root:

- type: Object
- properties: "config"

property	type	Value
config	Object	<u>properties</u> : "key", "value"
key	String	setting name (key)
value	String or Array	value

Example :

GET	/api/v1/admin/config?key=SMTP
Response : HTTP 200	
<pre>{ "config": { "key" : "SMTP", "value" : "1.2.3.4" } }</pre>	

GET	/api/v1/admin/config?key=ARRAY_CFG
Response : HTTP 200	
<pre>{ "config": { "key" : "ARRAY_CFG", "value" : ["5", "something"] } }</pre>	

Update

Request Document: JSON doc

root:

- type: Object with keys - keys to be stored and values to be stored. Call supports simple values and arrays.

property	type	Value
key	String	key
value	String or Array	value

```
{
  "BIOS_SMTP_SERVER" : "server",
  "BIOS_SMTP_VERIFY_CA" : true,
  "BIOS_SMTP_PORT" : 42,
  "BIOS_SNMP_COMMUNITY" : [ "ham", "spam" ]
}
```

or legacy version

```
{
  "config" : {
    "key" : "<key>",
    "value" : [ "<value1>", "<value2>", ... ]
  }
}
```

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)	Corresponding HTTP status code
-----------	------	------------------	--------------------------------

Success - Document : empty JSON doc

Example:

POST	/api/v1/admin/config
Request document:	
{ "SNMP_COMMUNITY": ["one", "two", "three"] }	
Response : HTTP 200	
{}	

systemctl

The `systemctl` REST API calls are an interface to management of the `systemd` units which wrap components of the product as OS services. These calls (and the related command-line wrappers `systemctl` and `journalctl`) limit the allowed actions and target `systemd` units to those provided directly by this project or which are intimately related third-party components.

list (Read)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/admin/systemctl/list	List available services	level 2

Response

Success - Document : JSON doc

root:

- type: Array
- items: String

property	type	Value
root:Array::item	String	systemd unit name

Example :

GET	/api/v1/admin/systemctl/list
Response : HTTP 200	
<pre>{ "systemctl_services" : ["service1", "service2", "service3"] }</pre>	

status (Read)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/admin/systemctl/status/<service_name>	Status of service	level 2

Request Parameters:

- **service_name:** supported service, target or timer unit name (Note: if a short "base name" of the unit without a supported extension is used, it will be externally expanded to the first matching unit name allowed for manipulation – if any – and that full name will be returned in the response documents)

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)	Corresponding HTTP status code
invalid <service_name>	TODO	Service <service_name> not supported.	404

Success - Document : JSON doc

root:

- type: Object
- properties: "<service_name>" (note: the FULL systemd unit name, with the .service, .target or .timer extension will be returned, if the request only provided a "base name" for the unit).

property	type	Value
<service_name>	Object	properties: "ActiveState", "SubState", "LoadState", "UnitFileState"
ActiveState	String	"active" / "inactive"
SubState	String	"running" / "dead"

LoadState	String	TODO
UnitFileState	String	"enabled" / "disabled"

Example :

GET	/api/v1/admin/systemctl/status/bios-agent-nut
Response : HTTP 200	
<pre>{ "bios-agent-nut" : { "ActiveState" : "active", "SubState" : "running", "LoadState" : "loaded", "UnitFileState" : "enabled" } }</pre>	

start (Update)

Operation	Method	URI	Description	Security
Update	POST	/api/v1/admin/systemctl/start	TODO	level 2

Request start of given service.

Request Document : JSON document

```
{ "service_name" : "<service_name>" }
```

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)	Corresponding HTTP status code
invalid or empty <operation> in 'admin/systemctl/<operation>'	TODO	Service <service_name> not supported or empty.	404
bad request document (empty, bad, invalid service_name ...)	TODO	TODO	400
problem starting <service_name>	TODO	The following <service_name> could not be started: <error>.	500

Success - Document : JSON doc

root:

- type: Object
- properties: "<service_name>"

property	type	Value
<service_name>	Object	properties: "ActiveState", "SubState", "LoadState", "UnitFileState"
ActiveState	String	"active" / "inactive"
SubState	String	"running" / "dead"
LoadState	String	TODO
UnitFileState	String	"enabled" / "disabled"

Example :

POST	/api/v1/admin/systemctl/start/
{ "service_name" : "bios-agent-nut" }	
Response : HTTP 200	
{ "bios-agent-nut" : { "ActiveState" : "<active_state>", "SubState" : "<sub_state>", "LoadState" : "<load_state>", "UnitFileState" : "<unit_file_state>" } }	

stop (Update)

Operation	Method	URI	Description	Security
Update	POST	/api/v1/admin/systemctl/stop	TODO	level 2

Request stop of given service.

Request Document : JSON document

{ "service_name" : "<service_name>" }

Response

Error - Document: Error message template

Condition	Code	Message(English)	Corresponding HTTP status code
invalid or empty <operation> in 'admin/systemctl/<operation>'	TODO	Service <service_name> not supported or empty.	404
bad request document (empty, bad, invalid service_name ...)	TODO	TODO	400
problem starting <service_name>	TODO	The following <service_name> could not be started: <error>.	500

Success - Document : JSON doc

root:

- type: Object
- properties: "<service_name>"

property	type	Value
<service_name>	Object	properties: "ActiveState", "SubState", "LoadState", "UnitFileState"
ActiveState	String	"active" / "inactive"

SubState	String	"running" / "dead"
LoadState	String	TODO
UnitFileState	String	"enabled" / "disabled"

Example :

POST	/api/v1/admin/systemctl/stop/
{ "service_name" : "bios-agent-nut" }	
Response : HTTP 200	
<pre>{ "bios-agent-nut" : { "ActiveState" : "<active_state>", "SubState" : "<sub_state>", "LoadState" : "<load_state>", "UnitFileState" : "<unit_file_state>" } }</pre>	

restart (Update)

Operation	Method	URI	Description	Security
Update	POST	/api/v1/admin/systemctl/restart	TODO	level 2

Request Document : JSON document

{ "service_name" : "<service_name>" }

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)	Corresponding HTTP status code
invalid or empty <operation> in 'admin/systemctl/<operation>'	TODO	Service <service_name> not supported or empty.	404
bad request document (empty, bad, invalid service_name ...)	TODO	TODO	400
problem starting <service_name>	TODO	The following <service_name> could not be started: <error>.	500

Success - Document : JSON doc

root:

- type: Object
- properties: "<service_name>"

property	type	Value
<service_name>	Object	properties: "ActiveState", "SubState", "LoadState", "UnitFileState"

ActiveState	String	"active" / "inactive"
SubState	String	"running" / "dead"
LoadState	String	TODO
UnitFileState	String	"enabled" / "disabled"

Example :

POST	/api/v1/admin/systemctl/restart/
{ "service_name" : "bios-agent-nut" }	
Response : HTTP 200	
<pre>{ "bios-agent-nut" : { "ActiveState" : "<active_state>", "SubState" : "<sub_state>", "LoadState" : "<load_state>", "UnitFileState" : "<unit_file_state>" } }</pre>	

enable (Update)

Operation	Method	URI	Description	Security
Update	POST	/api/v1/admin/systemctl/enable	TODO	level 2

Enables a service to be started on bootup.

Request Document : JSON document

```
{ "service_name" : "<service_name>" }
```

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)	Corresponding HTTP status code
invalid or empty <operation> in 'admin/systemctl/<operation>'	TODO	Service <service_name> not supported or empty.	404
bad request document (empty, bad, invalid service_name ...)	TODO	TODO	400
problem starting <service_name>	TODO	The following <service_name> could not be started: <error>.	500

Success - Document : JSON doc

root:

- type: Object
- properties: "<service_name>"

property	type	Value
<service_name>	Object	properties: "ActiveState", "SubState", "LoadState", "UnitFileState"

ActiveState	String	"active" / "inactive"
SubState	String	"running" / "dead"
LoadState	String	TODO
UnitFileState	String	"enabled" / "disabled"

Example :

POST	/api/v1/admin/systemctl/enable/
{ "service_name" : "bios-agent-nut" }	
Response : HTTP 200	
<pre>{ "bios-agent-nut" : { "ActiveState" : "<active_state>", "SubState" : "<sub_state>", "LoadState" : "<load_state>", "UnitFileState" : "<unit_file_state>" } }</pre>	

disable (Update)

Operation	Method	URI	Description	Security
Update	POST	/api/v1/admin/systemctl/disable	TODO	level 2

Disables a service to not start during bootup.

Request Document: JSON document

```
{ "service_name" : "<service_name>" }
```

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)	Corresponding HTTP status code
invalid or empty <operation> in 'admin/systemctl/<operation>'	TODO	Service <service_name> not supported or empty.	404
bad request document (empty, bad, invalid service_name ...)	TODO	TODO	400
problem starting <service_name>	TODO	The following <service_name> could not be started: <error>.	500

Success - Document : JSON doc

root:

- type: Object
- properties: "<service_name>"

property	type	Value
<service_name>	Object	properties: "ActiveState", "SubState", "LoadState", "UnitFileState"

ActiveState	String	"active" / "inactive"
SubState	String	"running" / "dead"
LoadState	String	TODO
UnitFileState	String	"enabled" / "disabled"

Example :

POST	/api/v1/admin/systemctl/disable/
{ "service_name" : "bios-agent-nut" }	
Response : HTTP 200	
<pre>{ "bios-agent-nut" : { "ActiveState" : "<active_state>", "SubState" : "<sub_state>", "LoadState" : "<load_state>", "UnitFileState" : "<unit_file_state>" } }</pre>	

systemctl.networking

The **networking** service, which behaves differently

1. Only **show** and **restart** commands are available, otherwise BAD REQUEST is returned
2. When making **restart**, OK is returned immediately. Actual execution of restart is done asynchronously outside of HTTP server context to ensure no manipulation with network will cause this script to be killed by the system. The actual restart of networking is non trivial operation and can take **several minutes** to be finished!

Managing networks via /admin/systemctl

netcfgs (Read)

Operation	Method	URI	Description
Read	GET	/api/v1/admin/netcfgs	Retrieves list of configurable network interfaces

Security : Public access (level 0)

Example :

GET	/api/v1/admin/netcfgs
Response : HTTP 200	
<pre>{ "netcfgs": ["eth0", "eth1", "eth2"] }</pre>	

netcfg (Read, Update)

Operation	Method	URI	Description
Read	GET	/api/v1/admin/netcfg/{iface}	Retrieves settings of specified interface
Update	PUT	/api/v1/admin/netcfg/{iface}	Updates settings of specified interface

Read

Security : Public access (level 0)

Example :

GET	/api/v1/admin/netcfg/eth1
Response : HTTP 200	
<pre>{ "eth1": { "method": "static", "address": "192.168.1.1" , "netmask": "255.255.255.0", "gateway": "192.168.1.254", "nameservers": ["8.8.8.8", "8.8.4.4"] }</pre>	

GET	/api/v1/admin/netcfg/eth0
Response : HTTP 200	
<pre>{ "eth0": { "method": "dhcp" }</pre>	

Update

Security : Complete access (level 2)

This REST API call only changes corresponding config file. In order for these changes to take effect the networking service must be restarted. We explicitly do not specify who is responsible for networking service restart (since the REST API call does not know). Reboot of the system also triggers a restart of the networking service.

Format of data of this PUT request: JSON

```
{
  "<interface_name>" : {
    "<setting>" : <value>
  }
}
```

where

- <interface_name> must be a valid interface (i.e. one from /api/v1/admin/netcfgs output);
- <setting> can be one of the following strings: `method`, `address`, `netmask`, `gateway`, `nameservers`
- <value> is of type string except for `nameservers` when it's an array of strings

The following checks are performed:

- <value> is checked for validity of supplied IP addresses for `address`, `netmask`, `gateway`, `nameservers`. Only IPV4 addresses are supported. IPV6 is NOT supported.
- <value> can contain the following strings for `method`: `static`, `dhcp`.
- if `method` is not `static` then presence of settings `address`, `netmask`, `gateway` trigger an error.
- if `method` is being changed from `static` then any possible settings of `address`, `netmask`, `gateway` are deleted.

Examples:

PUT	/api/v1/admin/netcfg/eth1
-----	---------------------------

```
{
  "eth1": {
    "address": "192.168.2.2" ,
    "netmask": "255.255.255.128",
    "gateway": "192.168.2.1",
    "nameservers": [ "192.168.2.1" ]
  }
}
```

Response : HTTP 200

```
{
  "eth1": {
    "method": "static",
    "address": "192.168.2.2",
    "netmask": "255.255.255.128",
    "gateway": "192.168.2.1",
    "nameservers": [ "192.168.2.1" ]
  }
}
```

PUT	/api/v1/admin/netcfg/eth1
<pre>{ "eth1": { "method": "dhcp" } }</pre>	
Response : HTTP 200	
<pre>{ "eth1": { "method": "dhcp", "nameservers": ["192.168.2.1"] } }</pre>	

`method` for "eth1" is now `dhcp`:

PUT	/api/v1/admin/netcfg/eth1
<pre>{ "eth1": { "address": "192.168.2.2" , "netmask": "255.255.255.128", "gateway": "192.168.2.1", "nameservers": ["192.168.2.4"] } }</pre>	
Response : HTTP 400	

ifaces (Read)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/admin/ifaces	Retrieves list of network interfaces	level 0

Retrieves a list of network interfaces from the system.

The difference between 'admin/ifaces' and 'admin/netcfgs' rest api calls is that the former retrieves network interfaces that are being used in the system at the moment while the latter retrieves what is stored in the config file (waiting for a service reload for example).

Response

Error - Document: [Error message template](#)

Success - Document : JSON doc

root:

- type: Object
- properties: "ifaces"

property	type	Value
ifaces	Array	
ifaces::Array::items	String	names of network interfaces

Example :

GET	/api/v1/admin/ifaces
Response : HTTP 200	
{ "ifaces": ["eth0", "eth1", "eth2"] }	

iface (Read)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/admin/iface/<iface_name>	Retrieves settings of specified interface	level 0

Request Parameters:

- **iface_name**: valid interface name

The difference between 'admin/ifaces' and 'admin/netcfgs' rest api calls is that the former retrieves network interfaces that are being used in the system at the moment while the latter retrieves what is stored in the config file (waiting for a service reload for example).

Response

Error - Document: [Error message template](#)

Condition	Code	Message(English)	Corresponding HTTP status code
<iface_name> empty	TODO	interface name required.	400
invalid <iface_name>	TODO	Interface <iface_name> does not exist.	400

Success - Document : JSON doc

root:

- type: Object
- properties: "<iface>"

property	type	Value
<iface>	Object	<u>properties</u> : "state", "ip", "gateway", "link", "mac", "nameservers"
state	String	state of interface: "up", "down"
ip	Array	

ip::Array::items	Object	properties: "address", "netmask"
ip::Array::items::address	String	ip address
ip::Array::items::netmask	String	netmask
gateway	String	default gateway
link	String	Link detected: "yes", "no", "unknown"
mac	String	MAC address of the NIC
nameservers	Array	
nameservers::Array::items	String	nameservers

Example :

GET	/api/v1/admin/iface/eth0
Response : HTTP 200	
<pre>{ "eth0": { "state": "up", "ip" : [{ "address" : "10.130.38.192", "netmask": "255.0.0.0" }, { "address" : "1.2.3.4", "netmask" : "255.255.0.0" }], "gateway" : "10.130.38.2", "link" : "yes", "mac" : "fc:15:b4:eb:65:ce", "nameservers" : ["8.8.8.8", "4.4.4.4"] } }</pre>	

getlog (Read)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/admin/getlog	Returns a JSON document with the list of supported "lognames" and extensions, which this REST API call can serve. Example output (corresponding to server-side implementation capabilities): <pre>{ "getlog-supports": { "logname_base": ["messages"], "logname_ext": ["", ".txt", ".gz"] } }</pre>	level 2
Read	GET	/api/v1/admin/getlog/<logname>	Returns uncompressed text of the allowed log file (it is up to the browser how to display this `text/plain` data)	level 2
Read	GET	/api/v1/admin/getlog/<logname>.txt	Returns uncompressed text of the allowed log file (including HTTP filename headers to allow the browser to save it as a download)	level 2
Read	GET	/api/v1/admin/getlog/<logname>.gz	Returns gzip-compressed text of the allowed log file (including HTTP filename headers to allow the browser to save it as a download)	level 2

The `/api/v1/admin/getlog` operation allows the authenticated admin to view or download current logs, specifically those from an allowed list of the "lognames". The REST API call accepts the target "logname" and one of the supported optional extensions by mapping components of the natural requested URI; there are no request arguments (as the `?key=value&...` list separated from the base URI).

Currently the "messages" logname (internally mapped to the active copy of `/var/log/messages` file) is permitted; while the rotated-away copies of the log are not accessible through this call at this time. At later revisions, more lognames may get supported, and not all necessarily backed by pre-existing files (e.g. an export from in-memory cache of `system-journal` may be possible). An authoritative list of supported lognames is served by requesting the base URL (without a trailing slash!)

Specifying the logname without an extension causes its uncompressed plaintext to be displayed inline in the browser window itself; a present extension acts as a modifier to the action, currently supported are:

- a `.txt` extension to return the same uncompressed plaintext with an additional HTTP response header to facilitate downloading the log

- as a text file (marked with logname, hostname of the rack controller, and the current timestamp) and the `text/plain` content-type;
- a `.gz` extension to return the gzip-compressed stream with an additional HTTP response header to facilitate downloading the log as a `.gz` file (marked with logname, hostname of the rack controller, and the current timestamp) and the `application/gzip` content-type.

Several detected error conditions cause the HTTP code to be set as an error, with details returned as a JSON document instead. Depending on situation (and limitations of the tntnet webserver engine) the `application/json` content-type may be or not be set in these cases.

Condition	HTTP code	code	message
Unsupported "logname" was passed (by an user otherwise authorized to make this REST API call)	400/Bad request	47	Parameter 'logname' has bad value. Received %s. Expected 'messages' optionally with '.gz' or '.txt' extension
Error while reading or compressing the log file data (also a fallback code path for supported but not implemented extensions or lognames during development). Details about the specific error case are placed in the <code>webserver.log</code> .	500/Internal Error	42	Internal Server Error. Exception caught. Please check logs for more details.
Called with bad method (other than GET)	405/Method Not Allowed	45	Http method '\$method' is not allowed.
Cannot authorize user	407/Not Authorized	43	You are not authorized. Please use <code>'/oauth2/token?username=<user_name>&password=<password>&grant_type=password'</code> GET request to authorize.

license (Read, Update)

Operation	Method	URI	Description
Read	GET	<code>/api/v1/admin/license</code>	Returns text of the latest license agreement
Read	GET	<code>/api/v1/admin/license/status</code>	Returns status of license acceptance
Update	POST	<code>/api/v1/admin/license</code>	Accept current license agreement

Condition	HTTP code	code	message
Cannot read internal license document	500/Internal Error	42	Internal Server Error. Error reading license file, check integrity of storage.
Can't get user name from user id	500/Internal Error	42	Internal Server Error. Can't get user name for user with id \$uid.
Can't save the license	500/Internal Error	42	Error saving license acceptance or getting license version, check integrity of storage.
Called with bad method (other than GET/POST)	405/Method Not Allowed	45	Http method '\$method' is not allowed.
Cannot authorize user	407/Not Authorized	43	You are not authorized. Please use <code>'/oauth2/token?username=<user_name>&password=<password>&grant_type=password'</code> GET request to authorize.

passwd (Update)

Operation	Method	URI	Description
Update	POST	<code>/api/v1/admin/passwd</code>	Changes user's password

POST	<code>/api/v1/admin/passwd</code>
<pre>{ "user": "user-name", "old_passwd" : "old-password", "new_passwd" : "new-password" }</pre>	
Response: HTTP 200	
<pre>{ "passwd" : "password for USER_NAME changed" }</pre>	

Condition	HTTP code	code	message
Fail to parse input document	400/Bad Request	48	Request document has invalid syntax.
One of field user, old_passwd or new_passwd empty	400/Bad Request	47	Parameter '\$param' has bad value. Received '<empty>'. Expected <user name or password>.
User is root	400/Bad Request	47	Parameter 'root' has bad value. Received 'root'. Expected <user name other than root>.
Can't authenticate user using old_passwd through saslauthd	400/Bad Request	47	Parameter '(user old_password)' has bad value. Received '<invalid user or password>'. Expected <valid user and password>.
Change of password to new_passwd failed	400/Bad Request	47	Parameter 'new_passwd' has bad value. Received 'password with problems: \$reason'. Expected <valid password>. Where \$reason will be free form string - for instance "it is too simple, it is too short".
Can't authenticate user with new password	500/	42	Internal Server Error. Can't authenticate user with new password, saslauthd service failed.

sysinfo (Read)

Operation	Method	URI	Description	Security
Read	GET	/api/v1/admin/sysinfo[?detail=yes]	Retrieves various system informations	Level 0 for basic info Level 2 for full info

Request parameters:

`detail=yes` (Optional) - provides additional information about the installed packages and running software (which are not necessarily the same thing) in one transaction, mostly aimed at field support queries (end-users should have a minimal amount of steps to do and make an error at, to return all the data we need to help them) and identification of various operating environments during development and validation (both manual and automated). In this aspect this is not purely a REST API call, since it changes nothing in the system, and returns a lot of different data items as a response to a single query.

Without extra arguments this call only returns the minimal build-identification information needed for Web-UI footer displayed on each page.

Note that this is one service which is exempt from the authentication requirements in the REST API server settings, and internally varies its behavior based on the detected web-user authentication level (if any). This aspect is also used in CI testing to validate that lack or presence of authentication does indeed impact programmatic behavior of the server side (outputs are considerably different for same query URLs).

Response Document : JSON doc

Any access level without a `?detail=yes` URL parameter provides as little information as needed to undersign each WebUI page footer:

Property name	type	mandatory	level	description
restapi-metadata	Object	Yes	0	The additional object <code>restapi-metadata</code> can contain sub-objects with build-time details about the binary/library that serves the REST API (the <code>bio s-core</code> repository), based on the information available during the compilation. Note that its information may differ from that reported in the <code>packages/package-name=core</code> on development and CI testing systems, because the developer's custom-compiled code does not have to match locally available packages (if any are installed at all). Note that some of these sub-objects may be not present, based on authorization level and availability of this data at compile-time.
restapi-metadata/release-details	Object	No	2	Container for identification of low-level system components relevant for validation and field support of this release of the appliance.
restapi-metadata/release-details/osimage-name	String	No*	2	Basename (without directory path and archiving extension) of the mounted OS image archive, if such details were at all detected, or a valid string like "OS image name is not available" otherwise. Example value: "deploy-image-16.01.28-20.14.43_armv71"

Level 0 with `?detail=yes` URL parameter:

Property name	type	mandatory	level	description
---------------	------	-----------	-------	-------------

operating-system	Object	Yes	0	Namespace for features of the operating environment where this copy of the REST API server is running
operating-system/container	String	Yes	0	Container technology used (lxc, lxc-libvirt, openvz, none, ...) or "N/A" if no specific value was successfully detected
operating-system/hypervisor	String	Yes	0	Hypervisor technology (oracle, vmware, libvirt-d-kvm, kvm, none, ...) or "N/A" if no specific value was successfully detected
operating-system/installation-date	String	Yes	0	Installation (license-acceptance) date in extended ISO8601 format like "2015-08-13T18:33:27Z" (or "License not accepted yet" otherwise)
operating-system/location	String	Yes	0	Address of one datacenter, if available (or "N/A - Error retrieving location" otherwise)
operating-system/uname	Object	Yes	0	Sub-namespace with `uname` results
operating-system/uname/sysname	String	Yes	0	Operating system name (e.g., "Linux")
operating-system/uname/nodename	String	Yes	0	Name within "some implementation-defined network", e.g. "bios-rc-demo"
operating-system/uname/release	String	Yes	1	Operating system release (e.g., "3.18.0") or "unauthorized" for anonymous access
operating-system/uname/version	String	Yes	1	Operating system version (e.g. kernel build details like "#9 SMP Wed Feb 3 13:36:52 CET 2016") or "unauthorized" for anonymous access
operating-system/uname/machine	String	Yes	0	Hardware identifier (e.g. "x86_64", "armv7l")
restapi-metadata	Object	Yes	0	The additional object <code>restapi-metadata</code> can contain sub-objects with build-time details about the binary/library that serves the REST API (the <code>bios-core</code> repository), based on the information available during the compilation. Note that its information may differ from that reported in the <code>packages/package-name=core</code> on development and CI testing systems, because the developer's custom-compiled code does not have to match locally available packages (if any are installed at all). Note that some of these sub-objects may be not present, based on authorization level and availability of this data at compile-time.
restapi-metadata/core-package	Object	Yes	0	Container for build details of the running binary built from the <code>bios-core</code> repository, as defined when running the <code>./configure</code> script.
restapi-metadata/core-package/package-name	String	Yes	0	The <code>\$PACKAGE</code> value as defined by the <code>configure</code> script when building the binaries. Example value: "bios"; empty if not defined. Used in installation paths, some prompts, etc.
restapi-metadata/core-package/package-version	String	Yes	0	The <code>\$PACKAGE_VERSION</code> value as defined by the <code>configure</code> script when building the binaries. Example value: "0.1.1453880895~9a861b1"; empty if not defined. Helps identify sources of this build.
restapi-metadata/core-package/package-bugreport	String	Yes	0	The <code>\$PACKAGE_BUGREPORT</code> value as defined by the <code>configure</code> script when building the binaries. Example value: "EatonIPCOpenSource@Eaton.com"; empty if not defined. Contact for support, community, etc. about this package.
restapi-metadata/core-package/package-url	String	Yes	0	The <code>\$PACKAGE_URL</code> value as defined by the <code>configure</code> script when building the binaries. Example value: "http://bios.ftp.eaton.com"; empty if not defined. Project website or distribution download location.
restapi-metadata/core-package/vendor	String	Yes	0	The optional <code>\$PACKAGE_VENDOR</code> value as defined by the <code>configure</code> script when building the binaries. Example value: "Eaton"; empty if not defined. Not a standard <code>autoconf/automake</code> variable (may be not defined): if set, it should be the answer to the question "Who distributes this build as a product?"

Level 2 with `?detail=yes` URL parameter : includes "Level 0 with `?detail=yes` URL parameter" and exposes the following additional information (asterisk in the "mandatory" field below means whether a property in a non-mandatory object is mandatory when the object is indeed present):

Property name	type	mandatory	level	description
restapi-metadata/source-repo	Object	No	2	<p>Container for source-code identification of the running binary built from the bios-core repository, as defined in the Git SCM details.</p> <p>This branch is (in the entirety detailed below) only visible to Power-User or higher authorization, and only if Git details were available at build time.</p>
restapi-metadata/source-repo/scm	String	Yes*	2	Hardcoded to "git" at this time.
restapi-metadata/source-repo/origin	String	Yes*	2	URL of the Git repository. Example value: "http://stash.mbt.lab.etn.com/scm/BIOS/core.git"
restapi-metadata/source-repo/branch	String	Yes*	2	Branch in the Git repository. Example value: "master"
restapi-metadata/source-repo/commit	String	Yes*	2	Commit ID (long hash) picked as HEAD for the build. Example value: "9a861b148c0532e06fb42497c30c94f5c58f9963"
restapi-metadata/source-repo/commit-timestamp	String	Yes*	2	<p>Timestamp of the commit picked as HEAD for the build, in seconds-count since the Unix Epoch (1970-01-01 UTC). Example value: "1453880895"</p> <p>(Roughly, the latest modification time of the source code before the build was made)</p>
restapi-metadata/source-repo/commit-timestamp-iso8601	String	Yes*	2	Timestamp of the commit picked as HEAD for the build, in ISO8601 format for the "Zulu" (UTC) time zone. Example value: "20160127T074815Z"
restapi-metadata/source-repo/status	String	Yes*	2	The `git status -q` output for the build; should be empty in production builds. Example value: "\? \? binary-not-gitignored\n" (means this file existed and was not filtered away by the .gitignore settings).
restapi-metadata/build-info	Object	No	2	<p>Container for build-system identification (what host, and when, compiled this running binary).</p> <p>This branch is only visible to Power-User or higher authorization, and only if either build-host or build-time details (or both) were detected at build time.</p>
restapi-metadata/build-info/build-host-name	String	No*	2	Hostname of the machine which built this running binary, if build-host details were at all detected. Example value: "obs"
restapi-metadata/build-info/build-host-os	String	No*	2	Operating system of the machine which built this running binary, if build-host details were at all detected. Example value: "Linux 3.16.7-29-default #1 SMP Fri Oct 23 00:46:04 UTC 2015 (6be6a97)"
restapi-metadata/build-info/build-host-uname	String	No*	2	The whole `uname` output on the machine which built this running binary, if build-host details were at all detected. Example value: "Linux obs 3.16.7-29-default #1 SMP Fri Oct 23 00:46:04 UTC 2015 (6be6a97) x86_64 x86_64 x86_64 GNU/Linux"
restapi-metadata/build-info/build-timestamp	String	No*	2	Timestamp of the compilation event (when the ./configure script was executed), in seconds-count since the Unix Epoch (1970-01-01 UTC). Example value: "1454494121", if build-time details were at all detected.
restapi-metadata/build-info/build-timestamp-iso8601	String	No*	2	Timestamp of the compilation event (when the ./configure script was executed), in ISO8601 format for the "Zulu" (UTC) time zone. Example value: "20160203T100841Z", if build-time details were at all detected.
restapi-metadata/release-details	Object	No	2	<p>Container for identification of low-level system components relevant for validation and field support, including (if present): u-Boot and ulmage partition sizes and checksums (for known data payload size and/or padded with 0xFF's to partition size); the modules archive and OS image locations, filenames and checksums; bios-core and bios-web commit ID and timestamp; bios-web and OS image "tarballing" timestamp (when the image was built). Values which were not detected or are not applicable to the current platform are empty strings; values not yet supported in a given release of the product are absent.</p> <p>This branch is only visible to Power-User or higher authorization, and specific items are only published if available at run time (in the worst case, this object would be empty and/or would contain empty string contents for certain sub-objects).</p> <p>On an RC3 or on a container deployed with the ci-reset-virtual-machine.sh script this information should also be available in an /etc/release-details.json file generated at boot-time (see e.g. BIOS-1964 and BIOS-2887).</p>
restapi-metadata/release-details/osimage-lsinfo	String	No*	2	<p>Location, size, modification time and other details returned by Linux ls program on the host (before chroot) of the mounted OS image archive, if such details were at all detected.</p> <p>Example value: "-rw-r--r-- 1 0 0 96534528 Jan 28 20:30 /mnt/nand/rootfs/deploy-image-16.01.28-20.14.43_armv7l.squashfs"</p>
restapi-metadata/release-details/osimage-filename	String	No*	2	<p>Full path on the host (before chroot) of the mounted OS image archive, if such details were at all detected.</p> <p>Example value: "/mnt/nand/rootfs/deploy-image-16.01.28-20.14.43_armv7l.squashfs"</p>

restapi-metadata/release-details/osimage-name	String	No*	2	<p>Basename (without directory path and archiving extension) of the mounted OS image archive, if such details were at all detected, or a valid string like "OS image name is not available" otherwise.</p> <p>Example value: "deploy-image-16.01.28-20.14.43_armv7l"</p>
restapi-metadata/release-details/osimage-cksum	String	No*	2	<p>Checksum (implicitly MD5 at the moment) of the mounted OS image archive, if such details were at all detected.</p> <p>Example value: "2f692180238d5aa7ed3ce68433282f72"</p>
restapi-metadata/release-details/osimage-build-ts	String	No*	2	<p>Build ("tarballing") timestamp of the mounted OS image archive, if such details were at all detected (from <code>/usr/share/bios-web/image-version.txt</code>). Should be in ISO8601 format per example below, if conversion from the original file contents were successful; otherwise can be the unconverted content or empty value.</p> <p>Example value: "20160330T183352Z"</p>
restapi-metadata/release-details/osimage-img-type	String	No*	2	<p>IMGTYP of the mounted OS image archive, if such details were at all detected (from <code>/usr/share/bios-web/image-version.txt</code>).</p> <p>Example values: "deploy", "devel", "epfl" etc.</p>
restapi-metadata/bios-release/modimage-lsinfo	String	No*	2	<p>Location, size, modification time and other details returned by Linux <code>ls</code> program on the host (before chroot) of the mounted modules archive, if such details were at all detected.</p> <p>Example value: "-rw-r--r-- 1 0 0 9515008 Apr 1 11:31 /mnt/nand/modules/3.18.0.squashfs"</p>
restapi-metadata/release-details/modimage-filename	String	No*	2	<p>Full path on the host (before chroot) of the mounted modules archive, if such details were at all detected.</p> <p>Example value: "/mnt/nand/modules/3.18.0.squashfs"</p>
restapi-metadata/release-details/modimage-cksum	String	No*	2	<p>Checksum (implicitly MD5 at the moment) of the mounted modules archive, if such details were at all detected.</p> <p>Example value: "faf25fc94443caee6d83576d527c8629"</p>
restapi-metadata/release-details/bios-core-commit-id	String	No*	2	<p>Commit ID (short hash) picked as HEAD for the build of "bios-core" REST API (same commit info as in <code>restapi-metadata/source-repo/commit</code>), if available at build-time or picked from <code>/usr/share/bios-web/git_details.txt</code>.</p> <p>Example value: "9a861b1"</p>
restapi-metadata/release-details/bios-core-commit-ts	String	No*	2	<p>Timestamp of the commit picked as HEAD for the build of "bios-core" REST API, in ISO8601 format for the "Zulu" (UTC) time zone (same commit info as in <code>restapi-metadata/source-repo/commit-timestamp-iso8601</code>), if available at build-time or picked from <code>/usr/share/bios-web/git_details.txt</code>.</p> <p>Example value: "20160127T074815Z"</p>
restapi-metadata/release-details/bios-web-commit-id	String	No*	2	<p>Commit ID (short hash) picked as HEAD for the build of Web-UI, if available from <code>/usr/share/bios-web/version.txt</code>.</p> <p>Example value: "a10faf5"</p>
restapi-metadata/release-details/bios-web-commit-ts	String	No*	2	<p>Timestamp of the commit picked as HEAD for the build of Web-UI, if available from <code>/usr/share/bios-web/version.txt</code>.</p> <p>Example value: "20160324T153631Z"</p>
restapi-metadata/release-details/bios-web-build-ts	String	No*	2	<p>Build ("tarballing") timestamp of the Web-UI, if available from <code>/usr/share/bios-web/version.txt</code>.</p> <p>Example value: "20160330T173319Z"</p>
restapi-metadata/release-details/uboot-commit-id-eaton	String	No*	2	<p>Commit ID (short hash) picked as HEAD for the build of "bios-boot" repository with our build recipes and possible tweaks or patches against upstream sources, if available at build-time of the u-Boot firmware image and subsequently during low-level boot.</p> <p>Example value: "9a861b1"</p>
restapi-metadata/release-details/uboot-commit-id-opengear	String	No*	2	<p>Commit ID (short hash) picked as HEAD for the upstream u-Boot sources from OpenGear, if available at build-time of the u-Boot firmware image and subsequently during low-level boot.</p> <p>Example value: "9a861b1"</p>
restapi-metadata/release-details/uboot-build-ts	String	No*	2	<p>Build timestamp of the u-Boot loader in the flashed partition, if available during low-level boot.</p> <p>Example value: "20160201T131135Z"</p>

restapi-metadata/release-details/u-boot-part-cksum	String	No*	2	<p>Checksum (implicitly MD5 at the moment) of the <code>u-Boot</code> loader in the flashed partition (only payload data size), if available during low-level boot.</p> <p>Example value: "41ac67939970360258e857b97920bfe4" or empty if not known / not calculated during boot.</p> <p>NOTE: Due to lack of reliable method of payload size detection, currently the <code>u-Boot</code> images are pre-padded to the fixed partition size and then checksummed – so these two sets of measurements are likely to be the same (or one checksum empty if pointless calculation was skipped).</p>
restapi-metadata/release-details/u-boot-part-bytes	String	No*	2	<p>Payload data size (in bytes) of the <code>u-Boot</code> loader in the flashed partition, if available during low-level boot.</p> <p>Example value: "3026540"</p>
restapi-metadata/release-details/u-boot-padded-cksum	String	No*	2	<p>Checksum (implicitly MD5 at the moment) of the <code>u-Boot</code> loader plus padding with 0xFF's in the flashed partition (whole partition size), if available during low-level boot.</p> <p>Example value: "41ac67939970360258e857b97920bfe4" or empty if not known / not calculated during boot.</p>
restapi-metadata/release-details/u-boot-padded-bytes	String	No*	2	<p>Partition size (in bytes) of the <code>u-Boot</code> loader, up to which we might pad, if available during low-level boot.</p> <p>Example value: "1048576"</p>
restapi-metadata/release-details/uimage-commit-id-eaton	String	No*	2	<p>Commit ID (short hash) picked as HEAD for the build of "bios-boot" repository with our build recipes and possible tweaks or patches against upstream sources, if available at build-time of the <code>uImage</code> firmware image and subsequently during low-level boot.</p> <p>Example value: "9a861b1"</p>
restapi-metadata/release-details/uimage-commit-id-opengear	String	No*	2	<p>Commit ID (short hash) picked as HEAD for the upstream Linux kernel and driver sources from OpenGear source code tree, if available at build-time of the <code>uImage</code> firmware image and subsequently during low-level boot.</p> <p>Example value: "9a861b1"</p>
restapi-metadata/release-details/uimage-build-ts	String	No*	2	<p>Build timestamp of the <code>uImage</code> kernel and miniroot in the flashed partition, if available during low-level boot.</p> <p>Example value: "20160201T131135Z"</p>
restapi-metadata/release-details/uimage-part-cksum	String	No*	2	<p>Checksum (implicitly MD5 at the moment) of the <code>uImage</code> kernel and miniroot in the flashed partition (only payload data size), if available during low-level boot.</p> <p>Example value: "dfb1e67c2854e565f7aadb0e16525cf" or empty if not known / not calculated during boot.</p>
restapi-metadata/release-details/uimage-part-bytes	String	No*	2	<p>Payload data size (in bytes) of the <code>uImage</code> kernel and miniroot in the flashed partition, if available during low-level boot.</p> <p>Example value: "3026540"</p>
restapi-metadata/release-details/uimage-padded-cksum	String	No*	2	<p>Checksum (implicitly MD5 at the moment) of the <code>uImage</code> kernel and miniroot plus padding with 0xFF's in the flashed partition (whole partition size), if available during low-level boot.</p> <p>Example value: "d7041762ef94ad519a52b29a131c170c" or empty if not known / not calculated during boot.</p>
restapi-metadata/release-details/uimage-padded-bytes	String	No*	2	<p>Partition size (in bytes) of the <code>uImage</code> kernel and miniroot, up to which we might pad, if available during low-level boot.</p> <p>Example value: "3145728"</p>
restapi-metadata/release-details/hardware-catalog-number	String	No*	2	<p>Catalog number (model name) of the hardware, if filled at the factory. May be empty on e.g. earlier units (development prototypes).</p> <p>Example value: "IPC3000E"</p>
restapi-metadata/release-details/hardware-spec-revision	String	No*	2	<p>Hardware specification revision of the device, if filled at the factory. May be empty on e.g. earlier units (development prototypes).</p> <p>Example value: "03"</p>
restapi-metadata/release-details/hardware-serial-number	String	No*	2	<p>Serial number of the hardware device, if filled at the factory. May be empty on e.g. earlier units (development prototypes); assumed to be unique otherwise.</p> <p>Example value: "LA71026010"</p>

packages	Array	Yes	2	Array of Objects Note that not always a simple concatenation of these substrings would produce the OS package name - e.g. package "core" with DPKG version "0.1.1432900871~f893959-1" would be represented by strings in the examples below (respin number is part of version, not commit-id).
package-name	String	Yes	2	Package name. Example value: "gcc-4.9-base:armhf" or "diffutils" or "core"
package-version	String	Yes	2	Package version in format "number" or "number~commit" (where "number" is defined by the OS package versioning conventions, may include a trailing "-respin Number"). Example value: "4.9.2-10" or "1:3.3-1+b1" or "0.1.1432900871-1"
commit	String	No	2	Commit-ID of the source version used for the build (if this information was available in packaging metadata). Example value: "f893959"

Example 1 : sysinfo call with level 2 access and detail=yes

GET	/api/v1/admin/sysinfo?detail=yes
Response : HTTP 200	
<pre>{ "operating-system": { "container": "N/A", "hypervisor": "N/A", "uname": { "sysname": "Linux", "nodename": "linux-hv6c.site", "release": "3.16.7-24-desktop", "version": "#1 SMP PREEMPT Mon Aug 3 14:37:06 UTC 2015 (ec183cc)", "machine": "x86_64" } }, "installation-date": "2015-12-08T11:59:02Z", "location": "110 Rue Blaise Pascal, 38330 Montbonnot-Saint-Martin" }, "packages": [{ "package-name": "acl", "package-version": "2.2.52-2", }, { "package-name": "adduser", "package-version": "3.113+nmu3", }, { "package-name": "alert-agent", "package-version": "0.0.1448620068", "commit": "46ea988-" }] }</pre>	

```
, {
  "package-name" : "apt",
  "package-version" : "1.0.9.8.1",
}
, {
  "package-name" : "augeas-lenses",
  "package-version" : "1.2.0-0.2",
}
, {
  "package-name" : "augeas-tools",
  "package-version" : "1.2.0-0.2",
}
, {
  "package-name" : "base-files",
  "package-version" : "8+deb8u2",
}
, {
  "package-name" : "base-passwd",
  "package-version" : "3.5.37",
}
, {
  "package-name" : "bash",
  "package-version" : "4.3-11+b1",
}
]
}
```

Example 2 : sysinfo call with detail=yes and intermediate restricted access (level 1); note that package "build-info" and "source-repo" information blocks are not visible without authorization, but the main process status is NOT "unauthorized", and more details about the operating system are available

GET	/api/v1/admin/sysinfo?detail=yes&access_token=QLP52iCyeEOSkeGPgtNPTSSyS0qw4N6WiuDdcwWZRv0=
-----	--

Response : HTTP 200

```
{
  "operating-system" : {
    "container": "unknown",
    "hypervisor": "unknown",
    "uname": {
      "sysname": "Linux",
      "nodename" : "linux-vrcj",
      "release": "3.11.10-21-desktop",
      "version": "#1 SMP PREEMPT Mon Jul 21 15:28:46 UTC 2014 (9a9565d)",
      "machine": "x86_64"
    }
  },
  "$BIOS": {
    "packages": [
      {
        "package-name": "core",
        "package-version": "0.1~1785220",
        "package-bugreport": "",
        "package-url": "",
        "vendor": "Eaton"
      }
    ],
    "main-process-status" : "unknown",
  }
}
```

Example 3 : sysinfo call with complete access (level 2) and detail=yes

GET	/api/v1/admin/sysinfo?detail=yes&access_token=QLP52iCyeEOSkeGPgtNPTSSyS0qw4N6WiuDdcwWZRv0=
------------	---

Response : HTTP 200

```
{
  "operating-system" : {
    "container": "unknown",
    "hypervisor": "unknown",
    "uname": {
      "sysname": "Linux",
      "nodename" : "linux-vrcj",
      "release": "3.11.10-21-desktop",
      "version": "#1 SMP PREEMPT Mon Jul 21 15:28:46 UTC 2014 (9a9565d)",
      "machine": "x86_64"
    }
  },
  "$BIOS": {
    "packages": [
      {
        "package-name": "core",
        "package-version": "0.1~1785220",
        "package-bugreport": "",
        "package-url": "",
        "vendor": "BIOS Packager"
      },
      {
        "source-repo": {
          "scm": "git",
          "origin": "../core-fork-jim",
          "branch": "feature/version-info",
          "commit": "1785220a00de25d85e19668b41f5840dd1429c5b",
          "commit-timestamp": "1421075728",
          "status": " M Makefile.am\n M configure.ac\n M src/web/src/sysinfo.ecpp\n M tools/builder.sh"
        },
        "build-info": {
          "build-host-name": "debian8",
          "build-host-os": "Linux 3.16.0-4-amd64 #1 SMP Debian 3.16.7-2 (2014-11-06)",
          "build-host-uname": "Linux debian8 3.16.0-4-amd64 #1 SMP Debian 3.16.7-2 (2014-11-06) x86_64 GNU/Linux",
          "build-timestamp": "1421088904"
        }
      }
    ]
  }
}
```

email/test (Create)

Test your SMTP connection.

Operation	Method	URI	Description
Setup	POST	/api/v1/admin/email/test?to=<email>	Sends test email to <email> recipient's address

Response Document : JSON doc

Property name	Type	Description
status	String	Status of the operation
error_code	String	Error code taken from Genepi project
reason	String	Reason of the failure

List of error codes with description (Genepi project):

- 1: if no recipient is specified
- 2: if the smtp server is unreachable
- 3: if the hostname couldn't be resolved

- 4: if the Authentication method is not supported
- 5: if Authentication failed
- 6: if SSL is not supported by smtp server
- 7: if the CA of the smtp server certificate isn't known by the card
- 8: if SSL is required by the smtp server
- 9: if sender address is not specified
- 10: if the reason is unknown

Example:

POST	/api/v1/admin/email/test?to=john@company.com
Response : HTTP 200	
<pre>"test status" : { "status" : "Succeeded", "error_code" : 0, "reason" : "" }</pre>	

email/feedback (Create)

Send a feedback email to Eaton support.

Operation	Method	URI	Description
Send	POST	/api/v1/admin/email/feedback	Sends feedback email

Query parameters

parameter	type	mandatory	description	example
reply	String	Yes	Email address where Eaton support can contact person	reply=cio@bigcorp.com
message	String	Yes	Body of email message to be sent	Wow, great product, Can I buy more?
context	String	Yes	UI context, from which page we got the feedback	/alerts
participate	String	No (defaults to false)	Indication if user is willing to participate in feedback	participate=yes
attach_system_state	bool	No (defaults to false)	If we want to attach system state with feedback email. Not yet implemented	N/A

Call accepts several files attached as multipart/form-data. Those will be attached to the email sent to Eaton.

Special email headers

- X-Eaton-IPC-uuid -unique identifier of each message
- X-Eaton-IPC-context - value of context string
- X-Eaton-IPC-machine-id - unique identifier of user machine (/etc/machine-id) if available
- X-Eaton-IPC-participate-in-feedback - yes means user is willing to participate
- X-Eaton-IPC-image-version - version of underlying image

email/vote

Send a voting email to Eaton.

Operation	Method	URI	Description
Send	POST	/api/v1/admin/email/vote	Sends feedback email

parameter	type	mandatory	description
value	number	Yes	Numeric value for a vote (0-3, ugly-amazing)
context	String	Yes	UI context, from which page we got the feedback

Special email headers

- X-Eaton-IPC-uuid -unique identifier of each message
- X-Eaton-IPC-context - value of context string
- X-Eaton-IPC-vote - numeric value of vote
- X-Eaton-IPC-machine-id - unique identifier of user machine (/etc/machine-id) if available

OAuth2 (v1)

token(Create)

There are two ways to obtain a new token : GET or POST method.

Status	Operation	Method	URI	Description
Deprecated	Create	GET	/api/v1/oauth2/token?username=<username>&password=<pwd>&gran_type=password	Retrieves a new authentication token

Version : 1

Security : Public access (level 0)

Request Parameters :

- **username** : user name
- **pwd**: password

Response Document : JSON doc

Property name	type	mandatory	description
access_token	string	Yes	token value
expires_in	number	Yes	The remaining lifetime of the access token.
token_type	"Bearer"	Yes	Bearer

Example :

GET	/api/v1/oauth2/token?username=smith&password=pwd1234&grant_type=password
Response : HTTP 200	
<pre>{ "access_token": "1/fFAGRnJru1FTz70BzhT3Zg", "expires_in": 3600, "token_type": "Bearer" }</pre>	

Status	Operation	Method	URI	Description
Actual	Create	POST	/api/v1/oauth2/token	Retrieves a new authentication token

Version : 1

Security : Public access (level 0)

Request Document : JSON document

```
{
  "username": "someusername",
  "password": "somepassword",
  "grant_type": "password"
}
```

Response Document : JSON doc

Property name	type	mandatory	description
access_token	string	Yes	token value
token_type	string	Yes	"bearer"
expires_in	number	Yes	The remaining lifetime of the access token in second.

Example :

POST	/api/v1/oauth2/token
<pre>{ "username": "smith", "password": "pwd1234", "grant_type": "password" }</pre>	
Response : HTTP 200	
<pre>{ "access_token": "sgvUnKhymjiVRAHfOrtX/TfZc0rhw9LY1tvGSE3H/Ok=", "token_type": "bearer", "expires_in": 3569 }</pre>	

Condition	HTTP code	code	message
Bad grant_type	400/Bad Request	47	Parameter 'grant_type' has bad value. Received '\$grant_type'. Expected 'password'.
Bad user or password	403/Not Authorized	43	You are not authorized. Please use '/oauth2/token?username=<user_name>&password=<password>&grant_type=password' GET request to authorize.

revoke (Update)

Operation	Method	URI	Description	Security
Create	POST	/api/v1/oauth2/revoke	Invalidates token	level 2

POST request has to contain application/x-www-form-urlencoded bode with token variable set to token to invalidate.

see [RFC 7009](#) for more details

Response

POST	/api/v1/oauth2/token
token=sgvUnKhymjiVRAHfOrtX/TfZc0rhw9LY1tvGSE3H/Ok=	
Response : HTTP 200	
{ "success": "Everything went well" }	

Success:

Returns HTTP 200 Ok if token was invalidated or was invalid.

Condition	HTTP code	code	message
No access token	400/Bad Request	47	Parameter 'access_token (for revoke)' is required.