opCore API V2

opCore Integration (API version 2, Node config/info/status)

opCore API V2 is supported in opCharts 4.0.9 and greater, for earlier versions of opCharts, please refer to opCore API V1

opCore provides access to common data shared through all Opmantek applications. Not all applications expose or use all parts of opCore, opCharts allows access to the functionality below.

When listing resources (viewing the index) the default behaviour is to show only what has been asked for, by default only showing the ID's and modifiers will build up the info requested. When an individual resource is requested (show) all info is provided by default and modifiers will pair down the info provided.

Authentication is required to access all methods below.

API Endpoint

All requests are made under the following base URL:

```
http[s]://server/omk/opCharts/v2/
```

Request Modifiers (query parameters)

The properties request modifier tells opCore which properties you would like listed, query limits the requested resources to only those that match all criteria given. This collection can paginated, it is also limited to 25 results by default (this could be subject to change in future opCharts versions) use the limit query parameter to request more nodes

Not all requests will use all request modifiers. Arrays / KVP's are url encoded JSON. E.G. javascript: urlencode(JSON.stringify(array)); perl: URI::Escape::uri_escape(encode_json($array));

Query Parameter	Possible Values
properties	Array of property names. If provided only the properties specified will be returned (instead of the whole document. eg: properties=["configuration.customer", "configuration.group"] By default the nodes UUID is returned by default if no properties are given
filter	Array of key=value pairs, but coded in an array. Applied to the list of results in the order they are given. If an application key is provided that will be applied first. eg: filter=["configuration.group","NMIS8","catchall.data.nodestatus": "reachable"] (which is "configuration.group"="NMIS8" AND "catchall.data.nodestatus"="reachable")
count	(int) 0 or 1, if set to 1 the results are returned paged including the link to the next paged document and the total results for the requested collection
page	(int) Which page of the requested document to returned
limit	(int) How many results are returned, defaults to 25

Examples of how to use the request modifiers can be found in the response blocks below. In general, the queries will look something like this:

```
GET HTTP://server/omk/opCharts/v2/nodes?limit=25&count=1&filter["configuration.group" : "DataCentre", "catchall.data.nodestatus": "reachable"]&page=1&properties=["configuration.customer", "configuration.group"]
```

This will query all nodes that are in the group NMIS8 and return their nodestatus':

```
{
    "nodes": [
        {
            "configuration": {
                "customer": "Opmantek",
                "group": "DataCentre"
            },
            "uuid": "3f49619e-b8ae-4e96-b56a-a7331baf71d3"
        }
    ],
    "total": 1
}
```

Nodes (Config/Status/Info)

opCore provides access to all known nodes in the system. Nodes can be listed or viewed individually.

Index (List)

`GET /nodes/`

Retrieves a list of all node id's/names (limited to 25 use the limit param to requst more), returns them as an an array with the key nodes. Accepts all modifiers. If property modifier is set, the results will be an array of objects (hashes) with the properties requested found inside the object (hash). This request will always return the uuid, along with any other info listed in properties.

Successful response

```
# no properties or query requested, limited to 25 nodes
# GET /omk/opCharts/v2/nodes/
{
    "nodes": [
        {
            "uuid": "44406ddf-8424-4f01-8314-9606911e5a05"
        },
        {
            "uuid": "7949befd-c70e-4921-ad52-ced66489783b"
        },
        {
            "uuid": "882d09cf-e0da-4153-a652-4ad1ea506d61"
        },
....

# Pagination page size set to 5 start at page 2
# GET /omk/opCharts/v2/nodes?limit=5&count=1&page=2
{
    "@nextLink": "/omk/opCharts/v2/nodes?limit=5&count=1&page=2&page=3",
    "nodes": [
        {
            "uuid": "4c52b0f9-2144-49c9-8d4d-11ab7c9dd814"
        },
        {
            "uuid": "9c8ed1a9-b88a-4a53-9cd3-bf26c14d8322"
        },
        {
            "uuid": "bdf1706a-7694-466c-9773-70327a117af4"
        },
        {
            "uuid": "6c1ba844-b7c1-4854-bef1-88b0b77c6385"
        },
        {
            "uuid": "3f0af422-6100-4c8a-a7f4-4634c116d499"
        }
    ],
    "total": 536
}

#Filtered nodes by group and current status
# GET/omk/opCharts/v2/nodes?filter=["configuration.group" : "DataCentre", "catchall.data.nodestatus":
```

```
"degraded" ]
{
    "nodes": [
        {
            "uuid": "a18b705c-db8e-4aa6-b5d4-ff637d2adbbe"
        },
        {
            "uuid": "1ef5e314-76ec-4f3a-84ab-f6d6f6460f86"
        },
        {
            "uuid": "058ef0de-e6b8-4475-8c0b-c13712242919"
        },
    ]
}

# properties only: ["name","configuration.active","configuration.customer", "configuration.group"]
# GET /omk/opCharts/v2/nodes/?properties=["name","configuration.active","configuration.customer",
"configuration.group"]
{
    "nodes": [
        {
            "configuration": {
                "active": 1,
                "customer": "Opmantek",
                "group": "NMIS9"
            },
            "name": "master-nine-local",
            "uuid": "44406ddf-8424-4f01-8314-9606911e5a05"
        },

# query and properties: query=["node_name","asgard"]&properties=["config.group"]
# GET /omk/opCharts/v2/nodes?filter=["configuration.group" : "DataCentre", "catchall.data.nodestatus":
"degraded"]&properties=["name","configuration.active","configuration.customer", "configuration.group"]
{
    "nodes": [
        {
            "configuration": {
                "active": 1,
                "customer": "",
                "group": "DataCentre"
            },
            "name": "eris",
            "uuid": "a18b705c-db8e-4aa6-b5d4-ff637d2adbbe"
        },
```

Show (Get)

`GET /nodes/#uuid`

Retrieves the specified node object (hash) and returns all known info about it (so it may be large), accepts properties request modifier. It is recommended to use the properties query modifier to narrow down the data returned to only the data required.

Successful Response

```
# no properties specified, all data is returned, this will have some common data structures and some specific
to the node / model
# GET /omk/opCharts/v2/nodes/UUID1
{
        "uuid": "$UUID1",
        "name": "node1"
        "configuration": { ... data from this nodes configuration section ... },
        "summary": { ... data from this nodes summary ... },
        "cluster_id" : { ... uuid of which opHA server the node belongs to}
        ...
}
# properties=["configuration.services", "uuid", "catchall.data.last_poll",  "catchall.data.nodestatus"]
# GET /omk/opCharts/v2/nodes/UUID1?properties=["configuration.services", "uuid", "catchall.data.last_poll",
"catchall.data.nodestatus"]
{
    "catchall": {
        "data": {
            "last_poll": 1572835884.17908,
            "nodestatus": "degraded"
        }
    },
    "configuration": {
        "services": [
            "http_server",
            "nmis cgi",
            "port80"
        ]
    },
    "uuid": "44406ddf-8424-4f01-8314-9606911e5a05"
}
```

Show Interface (Get)

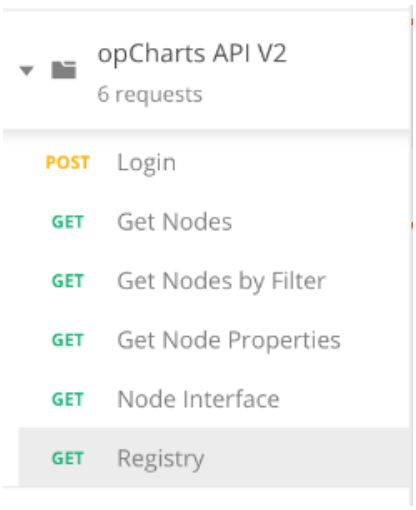
`GET /nodes/#uuid/interfaces/#index`

Retrieves the specified node interface object (hash) and returns all known info about it (so it may be large).

Successful Response

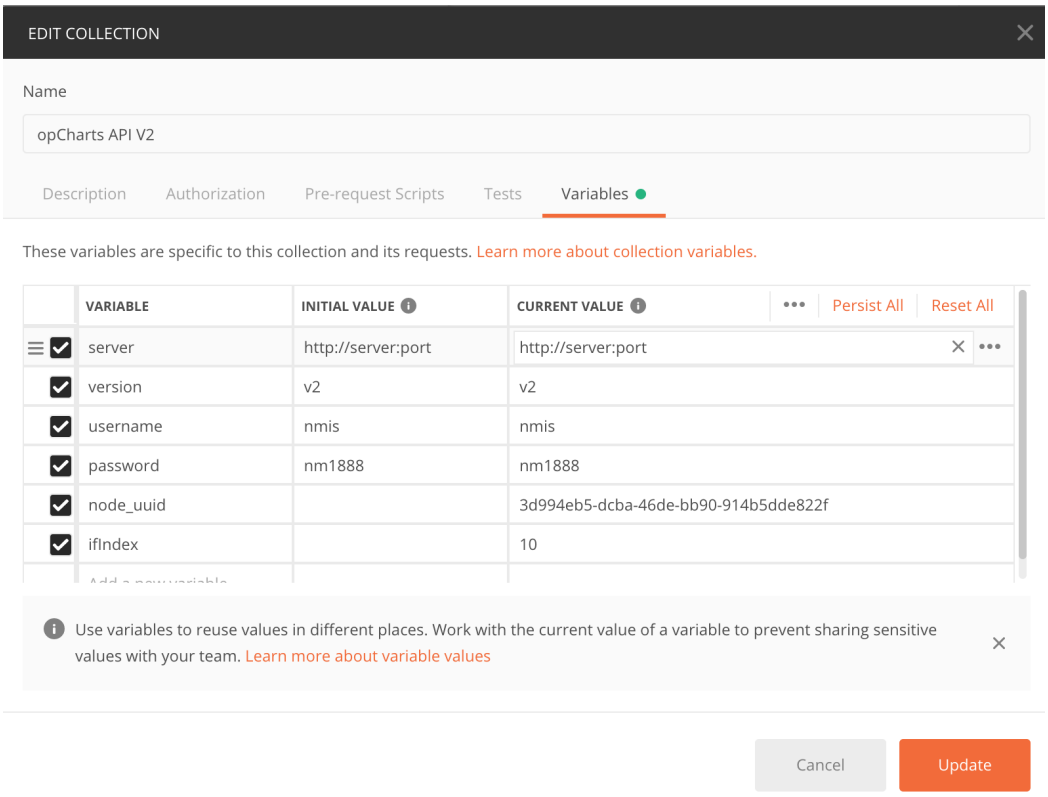
```
# GET /omk/opCharts/v2/nodes/UUID1/interfaces/IFINDEX
{
    "@nodes_interface_url": "/en/omk/opCharts/nodes/NODEUUID/resources/interface/indicies/10",
    "info": {
        "Description": "Connection to ...",
        "collect": "true",
        "event": "true",
        "ifAdminStatus": "up",
        "ifDescr": "Tunnel100",
        "ifHighSpeed": 0,
        "ifIndex": 10,
        "ifLastChange": "0:00:25",
        "ifLastChangeSec": 25,
        "ifOperStatus": "up",
        "ifPhysAddress": "",
        "ifSpeed": 9000,
        "ifType": "tunnel",
        "index": 10,
        "interface": "tunnel100",
        "ip": [
            {
                "ipAdEntAddr": "...",
                "ipAdEntNetMask": "...",
                "ipSubnet": "...",
                "ipSubnetBits": 30
            }
        ],
        "ipAdEntAddr1": "...",
        "ipAdEntNetMask1": "...",
        "ipSubnet1": "...",
        "ipSubnetBits1": 30,
        "nocollect": "Collecting: Collection Policy",
        "real": "true",
        "setlimits": "normal",
        "threshold": "true"
    },
    "name": "Eth1",
    "node": {
        "name": "nodeNAME",
        "uuid": "NODEUUID"
    },
    "status": {
        "discards_out": -1,
        "errors_in": -1,
        "util_in": 0,
        "util_out": 0.45
    }
}
```

Postman Collection



A postman collection is available to test all this methods easily.

Don't forget to edit the variables to set all the properties related to the environment:



Perl Example

Attached a Perl example code that accesses opCharts API V2 and generates a json file with some of the obtained data.

To test, you should update the URL, user and password properties.