Developers Guide

OpenNMS v16.0.3
Last updated 2015-06-10 10:58:55 EDT

Table of Contents

Chapter 1. CORS Support

1.1. Why do I need CORS support?

By default, many browsers implement a *same origin policy* which prevents making requests to a resource, on an origin that's different from the source origin.

For example, a request originating from a page served from http://www.opennms.org to a resource on http://www.adventuresinoss.com would be considered a cross origin request.

CORS (Cross Origin Resource Sharing) is a standard mechanism used to enable cross origin requests.

For further details, see:

- Mozilla's HTTP access control (CORS)
- W3C's CORS Spec

1.2. How can I enable CORS support?

CORS support for the REST interface (or any other part of the Web UI) can be enabled as follows:

1. Open '$OPENNMS_HOME/jetty-webapps/opennms/WEB-INF/web.xml' for editing.
2. Apply the CORS filter to the '/rest/' path by removing the comments around the <**filter-mapping**> definition. The result should look like:

```
<!-- Uncomment this to enable CORS support -->
<filter-mapping>
  <filter-name>CORS Filter</filter-name>
  <url-pattern>/rest/*</url-pattern>
</filter-mapping>
```

3. Restart OpenNMS

1.3. How can I configure CORS support?

CORS support is provided by the **org.ebaysf.web.cors.CORSFilter** servlet filter.

Parameters can be configured by modifying the filter definition in the 'web.xml' file referenced above.

By default, the allowed origins parameter is set to '*'.

The complete list of parameters supported are available from:

- https://github.com/ebay/cors-filter

Chapter 2. ReST API

A RESTful interface is a web service conforming to the REST architectural style as described in the book RESTful Web Services. This page is describes the RESTful interface for OpenNMS.

2.1. ReST URL

The base URL for Rest Calls is: http://opennmsserver:8980/opennms/rest/

For instance, http://localhost:8980/opennms/rest/alarms/ will give you the current alarms in the system.

2.2. Authentication

Use HTTP Basic authentication to provide a valid username and password. By default you will not receive a challenge, so you must configure your ReST client library to send basic authentication proactively.

2.3. Data format

Jersey allows ReST calls to be made using either XML or JSON. By default a request to the API is returned in XML. To get JSON encoded responses one has to send the following header with the request: `Accept: application/json`.

2.4. Standard Parameters

The following are standard params which are available on most resources (noted below)

Table 1. ReST standard parameter for resources

Parameter	Description
`limit`	integer, limiting the number of results. This is particularly handy on events and notifications, where an accidental call with no limit could result in many thousands of results being returned, killing either the client or the server. If set to 0, then no limit applied
`offset`	integer, being the numeric offset into the result set from which results should start being returned. E.g., if there are 100 result entries, offset is 15, and limit is 10, then entries 15-24 will be returned. Used for pagination
Filtering: All properties of the entity being accessed can be specified as parameters in either the *URL* (for *GET*) or the form value (for *PUT* and *POST*). If so, the value will be used to add a filter to the result. By default, the operation is equality, unless the `comparator` parameter is sent, in which case it applies to **all** comparisons in the filter. Multiple properties will result in an `AND` operation between the filter elements. Available comparators are:	
`eq`	Checks for equality
`ne`	Checks for non-equality
`ilike`	Case-insensitive wildcarding (`%` is the wildcard)
`like`	Case-sensitive wildcarding (`%` is the wildcard)
`gt`	Greater than
`lt`	Less than
`ge`	Greater than or equal
`le`	Less than or equal

If the value `null` is passed for a given property, then the obvious operation will occur (comparator will be ignored for that property). `notnull` is handled similarly.

- *Ordering*: If the parameter `orderBy` is specified, results will be ordered by the named property. Default is ascending, unless the `order` parameter is set to `desc` (any other value will default to ascending)

- *Raw where clause*: If there is a `query` parameter, it will be used as a raw where clause (*SQL*, not *HQL*), and added to any other filters created by other parameters

2.5. Standard filter examples

Take `/events` as an example.

Resource	Description
`/events?eventUei=uei.opennms.org/internal/rtc/subscribe`	would return the first 10 events with the rtc subscribe UEI, (10 being the default limit for events)
`/events?eventUei=uei.opennms.org/internal/rtc/subscribe&limit=0`	would return **all** the rtc subscribe events (potentially quite a few)
`/events?id=100&comparator=gt`	would return the first 10 events with an id greater than 100
`/events?eventAckTime=notnull`	would return the first 10 events that have a non-null Ack time (i.e. those that have been acknowledged)
`/events?eventAckTime=notnull&id=100&comparator=gt&limit=20`	would return the first 20 events that have a non-null Ack time and an id greater than 100. Note that the notnull value causes the comparator to be ignored for eventAckTime
`/events?eventAckTime=2008-07-28T04:41:30.530+12:00&id=100&comparator=gt&limit=20`	would return the first 20 events that have were acknowledged after 28th July 2008 at 4:41am (+12:00), and an id greater than 100. Note that the same comparator applies to both property comparisons.
`/events?orderBy=id&order=desc`	would return the 10 latest events inserted (probably, unless you've been messing with the id's)

2.6. Currently Implemented Interfaces

2.6.1. Acknowledgements

NOTE the default offset is 0, the default limit is 10 results. To get all results, use `limit=0` as a parameter on the URL (ie, `GET /acks?limit=0`).

GETs (Reading Data)

Resource	Description
`/acks`	Get a list of acknowledgements.
`/acks/count`	Get the number of acknowledgements. (Returns plaintext, rather than *XML* or *JSON*.)
`/acks/{id}`	Get the acknowledgement specified by the given *ID*.

POSTs (Setting Data)

Resource	Description
`/acks`	Creates or modifies an acknowledgement for the given alarm *ID* or notification *ID*. To affect an alarm, set an `alarmId<` parameter in the URL-encoded *POST* body; to affect a notification, set `notifyId` instead. An `action` parameter is also required, and may be one of `ack`, `unack`, `clear`, or `esc` (escalate).

Usage examples with curl

Acknowledge notification #3

```
curl -u 'admin:admin' -X POST -d notifId=3 -d action=ack http://localhost:8980/opennms/rest/acks
```

Escalate alarm #42

```
curl -u 'admin:admin' -X POST -d alarmId=42 -d action=esc http://localhost:8980/opennms/rest/acks
```

2.6.2. Alarm Statistics

It is possible to get some basic statistics on alarms, including the number of acknowledged alarms, total alarms, and the newest and oldest of acknowledged and unacknowledged alarms.

GETs (Reading Data)

Resource	Description
`/stats/alarms`	Returns statistics related to alarms. Accepts the same Hibernate parameters that you can pass to the `/alarms` ReST service.
`/stats/alarms/by-severity`	Returns the statistics related to alarms, one per severity. You can optionally pass a list of severities to the `severities` query parameter to limit it to the specified severities. (eg, `GET /opennms/rest/stats/alarms/by-severity?severities=MAJOR,CRITICAL`).

2.6.3. Alarms

NOTE	the default offset is 0, the default limit is 10 results. To get all results, use `limit=0` as a parameter on the URL (ie, `GET /events?limit=0`).

GETs (Reading Data)

Resource	Description
`/alarms`	Get a list of alarms.
`/alarms/count`	Get the number of alarms. (Returns plaintext, rather than *XML* or *JSON*.)
`/alarms/{id}`	Get the alarms specified by the given *ID*.

Note that you can also query by severity, like so:

Resource	Description
`/alarms?comparator=ge&severity=MINOR`	Get the alarms with a severity greater than or equal to *MINOR*.

PUTs (Modifying Data)

PUT requires form data using application/x-www-form-urlencoded as a Content-Type.

Resource	Description
`/alarms/{id}?ack=''(true;false)''`	Acknowledges (or unacknowledges) an alarm.
`/alarms?x=y&  &ack=''(true;false)''`	Acknowledges (or unacknowledges) alarms matching the additional query parameters. eg, `/alarms?node.id=4&ack=true`

New in OpenNMS 1.11.0

In OpenNMS 1.11.0, some additional features are supported in the alarm ack API:

Resource	Description
`/alarms/{id}?clear=true`	Clears an alarm.
`/alarms/{id}?escalate=true`	Escalates an alarm. eg, NORMAL MINOR, MAJOR CRITICAL, etc.
`/alarms?x=y&  &clear=true`	Clears alarms matching the additional query parameters.
`/alarms?x=y&  &escalate=true`	Escalates alarms matching the additional query parameters.

Additionally, when acknowledging alarms (ack=true) you can now specify an `ackUser` parameter. You will only be allowed to `ack` as a different user IF you are PUTting as an authenticated user who is in the *admin* role.

Queries

As noted above, it is possible to pass a raw `query` parameter when doing ReST queries. In the case of alarms, it is possible to pass severity names when querying by severity, rather than having to know the number that the severity enum maps to. For example:

```
/alarms?query=lastEventTime%20%3E%20'2011-08-19T11%3A11%3A11.000-
07%3A00'%20AND%20severity%20%3E%20MAJOR%20AND%20alarmAckUser%20IS%20NULL
```

This will get any alarms where the last event associated with the alarm is newer than August 19th, 2011 11:11:11, the severity is greater than *MAJOR*, and the alarm is not acknowledged (`alarmAckUser` is null). You should be able to use any column in the *alarm*, *event*, *node*, *ipinterface*, or *snmpinterface* tables.

2.6.4. Events

GETs (Reading Data)

Resource	Description
`/events`	Get a list of events. The default for offset is 0, and the default for limit is 10. To get all results, use limit=0 as a parameter on the URL (ie, `GET /events?limit=0`).
`/events/count`	Get the number of events. (Returns plaintext, rather than *XML* or *JSON*.)
`/events/{id}`	Get the event specified by the given *ID*.

PUTs (Modifying Data)

PUT requires form data using `application/x-www-form-urlencoded` as a Content-Type.

Resource	Description
`/events/{id}?ack=''(true;false)`	Acknowledges (or unacknowledges) an event.
`/events?x=y&&ack=''(true;false)`	Acknowledges (or unacknowledges) the matching events.

2.6.5. Foreign Sources

ReSTful service to the OpenNMS Provisioning Foreign Source definitions. Foreign source definitions are used to control the scanning (service detection) of services for SLA monitoring as well as the data collection settings for physical interfaces (resources).

This API supports CRUD operations for managing the Provisioner's foreign source definitions. Foreign source definitions are POSTed and will be deployed when the corresponding requisition gets imported/synchronized by Provisiond.

If a request says that it gets the "active" foreign source, that means it returns the pending foreign source (being edited for deployment) if there is one, otherwise it returns the deployed foreign source.

GETs (Reading Data)

Resource	Description
`/foreignSources`	Get all active foreign sources.
`/foreignSources/default`	Get the active default foreign source.
`/foreignSources/deployed`	Get the list of all deployed (active) foreign sources.
`/foreignSources/deployed/count`	Get the number of deployed foreign sources. (Returns plaintext, rather than *XML* or *JSON*.)
`/foreignSources/{name}`	Get the active foreign source named {name}.
`/foreignSources/{name}/detectors`	Get the configured detectors for the foreign source named {name}.
`/foreignSources/{name}/detectors/{detector}`	Get the specified detector for the foreign source named {name}.
`/foreignSources/{name}/policies`	Get the configured policies for the foreign source named {name}.
`/foreignSources/{name}/policies/{policy}`	Get the specified policy for the foreign source named {name}.

POSTs (Adding Data)

POST requires XML using application/xml as its Content-Type.

Resource	Description
`/foreignSources`	Add a foreign source.
`/foreignSources/{name}/detectors`	Add a detector to the named foreign source.
`/foreignSources/{name}/policies`	Add a policy to the named foreign source.

PUTs (Modifying Data)

PUT requires form data using application/x-www-form-urlencoded as a Content-Type.

Resource	Description
`/foreignSources/{name}`	Modify a foreign source with the given name.

DELETEs (Removing Data)

Resource	Description
`/foreignSources/{name}`	Delete the named foreign source.
`/foreignSources/{name}/detectors/{detector}`	Delete the specified detector from the named foreign source.
`/foreignSources/{name}/policies/{policy}`	Delete the specified policy from the named foreign source.

2.6.6. Groups

Like users, groups have a simplified interface as well.

GETs (Reading Data)

Resource	Description
`/groups`	Get a list of groups.
`/groups/{groupname}`	Get a specific group, given a group name.
`/groups/{groupname}/users`	Get the users for a group, given a group name. (new in OpenNMS 14)
`/groups/{groupname}/categories`	Get the categories associated with a group, given a group name. (new in OpenNMS 14)

POSTs (Adding Data)

Resource	Description
`/groups`	Add a new group.

PUTs (Modifying Data)

Resource	Description
`/groups/{groupname}`	Update the metadata of a group (eg, change the `comments` field).
`/groups/{groupname}/users/{username}`	Add a user to the group, given a group name and username. (new in OpenNMS 14)
`/groups/{groupname}/categories/{categoryname}`	Associate a category with the group, given a group name and category name. (new in OpenNMS 14)

DELETEs (Removing Data)

Resource	Description
`/groups/{groupname}`	Delete a group.
`/groups/{groupname}/users/{username}`	Remove a user from the group. (new in OpenNMS 14)
`/groups/{groupname}/categories/{categoryname}`	Disassociate a category from a group, given a group name and category name. (new in OpenNMS 14)

2.6.7. KSC Reports

GETs (Reading Data)

Resource	Description
`/ksc`	Get a list of all KSC reports, this includes ID and label.
`/ksc/{reportId}`	Get a specific KSC report, by ID.
`/ksc/count`	Get a count of all KSC reports.

PUTs (Modifying Data)

Resource	Description
`/ksc/{reportId}`	Modify a report with the given ID.

POSTs (Creating Data)

Documentation incomplete see issue: NMS-7162

DELETEs (Removing Data)

Documentation incomplete see issue: NMS-7162

2.6.8. Links

You can manipulate raw *Linkd DataLinkInterface* information using the links *API*.

GETs (Reading Data)

Resource	Description
`/links`	Get a list of links. The default for offset is 0, and the default for limit is 10. To get all results, use `limit=0` as a parameter on the URL (ie, `GET /links?limit=0`).
`/links/{id}`	Get a link specifically by *ID*.

PUTs (Modifying Data)

PUT requires form data using `application/x-www-form-urlencoded` as a Content-Type.

Resource	Description
`/links/{id}`	Modify an existing link.

POSTs (Creating Data)

{[options="header", cols="5,10"]

Resource	Description
`/links`	Add a new link.

DELETEs (Removing Data)

Resource	Description
`/links/{id}`	Delete the link with the given *ID*.

2.6.9. Maps

The *SVG maps* use *ReST* to populate their data. This is the interface for doing that.

GETs (Reading Data)

Resource	Description
`/maps`	Get the list of maps.
`/maps/{id}`	Get the map with the given *ID*.
`/maps/{id}/mapElements`	Get the elements (*nodes*, *links*, *etc.*) for the map with the given *ID*.

POSTs (Adding Data)

Resource	Description
`/maps`	Add a map.

PUTs (Modifying Data)

Resource	Description
`/maps/{id}`	Update the properties of the map with the given *ID*.

DELETEs (Removing Data)

Resource	Description
`/maps/{id}`	Delete the map with the given *ID*.

2.6.10. Measurements API

The *Measurements API* can be used to retrieve collected values stored in *RRD* (or *JRB*) files. Note that all units of time are expressed in milliseconds.

GETs (Reading Data)

Resource	Description
`/measurements/{resourceId}/{attribute}`	Retrieve the measurements for a single attribute

The following table shows all supported query string parameters and their default values.

name	default	comment
start	-14400000	Timestamp in milliseconds. If < 0, the effective value will be (end + start).
end	0	Timestamp in milliseconds. If 0, the effective value will be the current timestamp.

name	default	comment
step	300000	Requested time interval between rows. Actual step may differ. Set to 1 for maximum accuracy.
maxrows	0	When using the measurements to render a graph, this should be set to the graph's pixel width.
aggregation	AVERAGE	Consolidation function used. Can typically be `AVERAGE`, `MIN` or `MAX`. Depends on `RRA` definitions.

Usage examples with curl

Retrieve CPU counter metrics over the last 2 hours for node 1

```
curl -u admin:admin
"http://127.0.0.1:8980/opennms/rest/measurements/node%5B1%5D.nodeSnmp%5B%5D/CpuRawUser?start=-
7200000&maxrows=30&aggregation=AVERAGE"
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<query-response end="1425588138256" start="1425580938256" step="300000">
  <columns>
    <values>159.5957271523179</values>
    <values>158.08531037527592</values>
    <values>158.45835584842285</values>
    ...
  </columns>
  <labels>CpuRawUser</labels>
  <timestamps>1425581100000</timestamps>
  <timestamps>1425581400000</timestamps>
  <timestamps>1425581700000</timestamps>
  ...
</query-response>
```

POSTs (Reading Data)

Resource	Description
`/measurements`	Retrieve the measurements for one or more attributes, possibly spanning multiple resources, with support for JEXL expressions.

Here we use a POST instead of a GET to retrieve the measurements, which allows us to perform complex queries which are difficult to express in a query string. These requests cannot be used to update or create new metrics.

An example of the POST body is available bellow.

Usage examples with curl

Retrieve bits in and bits out metrics for a particular interface. Perform calculations on bits out, and only return the derived values.

```
curl -X POST  -H "Accept: application/json" -H "Content-Type: application/json" -u admin:admin  -d
@report.json  http://127.0.0.1:8980/opennms/rest/measurements
```

Contents of report.json

```
{
    "start": 1425563626316,
    "end": 1425585226316,
    "step": 10000,
    "maxrows": 1600,
    "source": [
        {
            "aggregation": "AVERAGE",
            "attribute": "ifHCInOctets",
            "label": "ifHCInOctets",
            "resourceId": "nodeSource[NODES:1424038123222].interfaceSnmp[eth0-04013f75f101]",
            "transient": "false"
        },
        {
            "aggregation": "AVERAGE",
            "attribute": "ifHCOutOctets",
            "label": "ifHCOutOctets",
            "resourceId": "nodeSource[NODES:1424038123222].interfaceSnmp[eth0-04013f75f101]",
            "transient": "true"
        }
    ],
    "expression": [
        {
            "label": "ifHCOutOctetsNeg",
            "value": "-1.0 * ifHCOutOctets",
            "transient": "false"
        }
    ]
}
```

Response


```
{
    "step": 300000,
    "start": 1425563626316,
    "end": 1425585226316,
    "timestamps": [
        1425563700000,
        1425564000000,
        1425564300000,
        ...
    ],
    "labels": [
        "ifHCInOctets",
        "ifHCOutOctetsNeg"
    ],
    "columns": [
        {
            "values": [
                139.94817275747508,
                199.0062569213732,
                162.6264894795127,
                ...
            ]
        },
        {
            "values": [
                -151.66179401993355,
                -214.7415503875969,
                -184.9012624584718,
                ...
            ]
        }
    ]
}
```


2.6.11. Nodes

Note: the default offset is 0, the default limit is 10 results. To get all results, use limit=0 as a parameter on the URL (ie, GET /nodes?limit=0).

Additionally, anywhere you use "id" in the queries below, you can use the foreign source and foreign ID separated by a colon instead (ie, GET /nodes/fs:fid).

GETs (Reading Data)

Resource	Description
`/nodes`	Get a list of nodes. This includes the ID and node label.
`/nodes/{id}`	Get a specific node, by ID.
`/nodes/{id}/ipinterfaces`	Get the list of IP interfaces associated with the given node.
`/nodes/{id}/ipinterfaces/{ipAddress}`	Get the IP interface for the given node and IP address.
`/nodes/{id}/ipinterfaces/{ipAddress}/services`	Get the list of services associated with the given node and IP interface.
`/nodes/{id}/ipinterfaces/{ipAddress}/services/{service}`	Get the requested service associated with the given node, IP interface, and service name.

Resource	Description
`/nodes/{id}/snmpinterfaces`	Get the list of SNMP interfaces associated with the given node.
`/nodes/{id}/snmpinterfaces/{ifIndex}`	Get the specific interface associated with the given node and ifIndex.
`/nodes/{id}/categories`	Get the list of categories associated with the given node.
`/nodes/{id}/categories/{categoryName}`	Get the category associated with the given node and category name.
`/nodes/{id}/assetRecord`	Get the asset record associated with the given node.

POSTs (Adding Data)

POST requires XML using application/xml as its Content-Type.

Resource	Description
`/nodes`	Add a node.
`/nodes/{id}/ipinterfaces`	Add an IP interface to the node.
`/nodes/{id}/ipinterfaces/{ipAddress}/services`	Add a service to the interface for the given node.
`/nodes/{id}/snmpinterfaces`	Add an SNMP interface to the node.
`/nodes/{id}/categories`	Add a category association to the node.

PUTs (Modifying Data)

PUT requires form data using application/x-www-form-urlencoded as a Content-Type.

Resource	Description
`/nodes/{id}`	Modify a node with the given ID.
`/nodes/{id}/ipinterfaces/{ipAddress}`	Modify the IP interface with the given node ID and IP address.
`/nodes/{id}/ipinterfaces/{ipAddress}/services/{service}`	Modify the service with the given node ID, IP address, and service name.
`/nodes/{id}/snmpinterfaces/{ifIndex}`	Modify the SNMP interface with the given node ID and ifIndex.
`/nodes/{id}/categories/{categoryName}`	Modify the category with the given node ID and name.

DELETEs (Removing Data)

Perform a DELETE to the singleton URLs specified in PUT above to delete that object.

2.6.12. Notifications

Note: the default offset is 0, the default limit is 10 results. To get all results, use `limit=0` as a parameter on the URL (ie, `GET /events?limit=0`).

GETs (Reading Data)

Resource	Description
`/notifications`	Get a list of notifications.
`/notifications/count`	Get the number of notifications. (Returns plaintext, rather than *XML* or *JSON*.)
`/notifications/{id}`	Get the notification specified by the given *ID*.

To acknowledge or unacknowledge a notification, use the `acks` endpoint — see Acknowledgements.

2.6.13. Outage Timelines

GETs (Reading Data)

Resource	Description
`/header/{start}/{end}/{width}`	Generate the timeline header
`/image/{nodeId}/{ipAddress}/{serviceName}/{start}/{end}/{width}`	Generate the timeline image
`/empty/{start}/{end}/{width}`	Generate an empty timeline for non-monitored services
`/html/{nodeId}/{ipAddress}/{serviceName}/{start}/{end}/{width}`	Generate the raw HTML for the image

2.6.14. Outages

GETs (Reading Data)

Resource	Description
`/outages`	Get a list of outages.
`/outages/count`	Get the number of outages. (Returns plaintext, rather than *XML* or *JSON*.)
`/outages/{id}`	Get the outage specified by the given *ID*.
`/outages/forNode/{nodeId}`	Get the outages that match the given node *ID*.

2.6.15. Requisitions

RESTful service to the OpenNMS Provisioning Requisitions. In this *API*, these "groups" of nodes are aptly named and treated as requisitions.

This current implementation supports *CRUD* operations for managing provisioning requisitions. Requisitions are first *POSTed* and no provisioning (import/synchronize) operations are taken. This is done so that a) the *XML* can be verified and b) so that the operations can happen at a later time. They are moved to the deployed state (put in the active requisition repository) when an import is run.

If a request says that it gets the *active* requisition, that means it returns the pending requisition (being edited for deployment) if there is one, otherwise it returns the deployed requisition. Note that anything that says it *adds/deletes/modifies* a *node*, *interface*, etc. in these instructions is referring to modifying that element from the *requisition* not from the database itself. That will happen upon import/synchronization.

You may write requisition data if the authenticated user is in the *provision*, *rest*, or *admin* roles.

GETs (Reading Data)

Resource	Description
`/requisitions`	Get all active requisitions.
`/requisitions/count`	Get the number of active requisitions. (Returns plaintext, rather than *XML* or *JSON*.)
`/requisitions/deployed`	Get the list of all deployed (active) requisitions.

Resource	Description
`/requisitions/deployed/count`	Get the number of deployed requisitions. (Returns plaintext, rather than *XML* or *JSON*.)
`/requisitions/{name}`	Get the active requisition for the given foreign source name.
`/requisitions/{name}/nodes`	Get the list of nodes being requisitioned for the given foreign source name.
`/requisitions/{name}/nodes/{foreignId}`	Get the node with the given foreign *ID* for the given foreign source name.
`/requisitions/{name}/nodes/{foreignId}/interfaces`	Get the interfaces for the node with the given foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}`	Get the interface with the given IP for the node with the specified foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}/services`	Get the services for the interface with the specified IP address, foreign *ID*, and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}/services/{service}`	Get the given service with the specified IP address, foreign *ID*, and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/categories`	Get the categories for the node with the given foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/categories/{categoryName}`	Get the category with the given name for the node with the specified foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/assets`	Get the assets for the node with the given foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/assets/{assetName}`	Get the value of the asset for the given assetName for the node with the given foreign *ID* and foreign source name.

POSTs (Adding Data)

Resource	Description
`/requisitions`	Adds (or replaces) a requisition.
`/requisitions/{name}/nodes`	Adds (or replaces) a node in the specified requisition. This operation can be very helpful when working with [[Large Requisitions]].
`/requisitions/{name}/nodes/{foreignId}/interfaces`	Adds (or replaces) an interface for the given node in the specified requisition.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}/services`	Adds (or replaces) a service on the given interface in the specified requisition.
`/requisitions/{name}/nodes/{foreignId}/categories`	Adds (or replaces) a category for the given node in the specified requisition.
`/requisitions/{name}/nodes/{foreignId}/assets`	Adds (or replaces) an asset for the given node in the specified requisition.

PUTs (Modifying Data)

Resource	Description
`/requisitions/{name}/import`	Performs an import/synchronize on the specified foreign source. This turns the "active" requisition into the "deployed" requisition.
`/requisitions/{name}/import?rescanExisting=false`	Performs an import/synchronize on the specified foreign source. This turns the "active" requisition into the "deployed" requisition. Existing nodes will not be scanned until the next rescan interval, only newly-added nodes will be. Useful if you're planning on making a series of changes.

Resource	Description
`/requisitions/{name}`	Update the specified foreign source.
`/requisitions/{name}/nodes/{foreignId}`	Update the specified node for the given foreign source.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}`	Update the specified IP address for the given node and foreign source.

DELETEs (Removing Data)

Resource	Description
`/requisitions/{name}`	Delete the pending requisition for the named foreign source.
`/requisitions/deployed/{name}`	Delete the active requisition for the named foreign source.
`/requisitions/{name}/nodes/{foreignId}`	Delete the node with the given foreign *ID* from the given requisition.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}`	Delete the IP address from the requisitioned node with the given foreign *ID* and foreign source.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}/services/{service}`	Delete the service from the requisitioned interface with the given IP address, foreign *ID* and foreign source.
`/requisitions/{name}/nodes/{foreignId}/categories/{category}`	Delete the category from the node with the given foreign *ID* and foreign source.
`/requisitions/{name}/nodes/{foreignId}/assets/{field}`	Delete the field from the requisition's nodes asset with the given foreign *ID* and foreign source.

2.6.16. Scheduled Outages

GETs (Reading Data)

Parameter	Description
`/sched-outages`	to get a list of configured scheduled outages.
`/sched-outages/{outageName}`	to get the details of a specific outage.

POSTs (Setting Data)

Parameter	Description
`/sched-outages`	to add a new outage (or update an existing one).

PUTs (Modifying Data)

Parameter	Description
`/sched-outages/{outageName}/collectd/{package}`	to add a specific outage to a collectd's package.

Parameter	Description
`/sched-outages/{outageName}/pollerd/{package}`	to add a specific outage to a pollerd's package.
`/sched-outages/{outageName}/threshd/{package}`	to add a specific outage to a threshd's package.
`/sched-outages/{outageName}/notifd`	to add a specific outage to the notifications.

DELETEs (Removing Data)

Parameter	Description
`/sched-outages/{outageName}`	to delete a specific outage.
`/sched-outages/{outageName}/collectd/{package}`	to remove a specific outage from a collectd's package.
`/sched-outages/{outageName}/pollerd/{package}`	to remove a specific outage from a pollerd's package.
`/sched-outages/{outageName}/threshd/{package}`	to remove a specific outage from a threshd's package.
`/sched-outages/{outageName}/notifd`	to remove a specific outage from the notifications.

2.6.17. SNMP Configuration

You can edit the community string, SNMP version, etc. for an IP address using this interface. If you make a change that would overlap with an existing snmp-config.xml, it will automatically create groups of <definition /> entries as necessary. If no <definition /> entry is created it matches the defaults.

There are different versions of the interface (see below). The following operations are supported:

GETs (Reading Data)

Parameter	Description
`/snmpConfig/{ipAddress}`	Get the SNMP configuration for a given IP address.

PUTs (Modifying Data)

Parameter	Description
`/snmpConfig/{ipAddress}`	Add or update the SNMP configuration for a given IP address.

Determine API version

To determine the version of the *API* running in your OpenNMS type http://localhost:8980/opennms/rest/snmpConfig/1.1.1.1 in your browser and have a look at the output:

- **Version 1:** If the output only have attributes `community`, `port`, `retries`, `timeout` and `version`
- **Version 2:** If there are more attributes than described before (e.g. max Repetitions)

API Version 1

In version 1 only a few attributes defined in `snmp-config.xsd` are supported. These are defined in `snmp-info.xsd`:

```
<xs:schema
    xmlns:tns="http://xmlns.opennms.org/xsd/config/snmp-info"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1.0"
    targetNamespace="http://xmlns.opennms.org/xsd/config/snmp-info">
  <xs:element name="snmp-info" type="tns:snmpInfo"/>
  <xs:complexType name="snmpInfo">
    <xs:sequence>
      <xs:element name="community" type="xs:string" minOccurs="0"/>
      <xs:element name="port" type="xs:int"/>
      <xs:element name="retries" type="xs:int"/>
      <xs:element name="timeout" type="xs:int"/>
      <xs:element name="version" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

The following table shows all supported attributes, optional restrictions and the mapping between `snmp-info.xsd` and `snmp-config.xsd`. All parameters can be set regardless the version.

attribute snmp-info.xml	attribute snmp-config.xml	default	restricted to version	restriction
version	version	v1	-	"v1", "v2c" or "v3" are valid arguments. If an invalid or empty argument is provided "v1" is used.
port	port	161	-	Integer > 0

attribute snmp-info.xml	attribute snmp-config.xml	default	restricted to version	restriction
retries	retry	1	-	Integer > 0
timeout	timeout	3000	-	Integer > 0
community	read-community	public	-	any string with a length >= 1

Example 1:

```
curl -v -X PUT -H "Content-Type: application/xml" \
     -H "Accept: application/xml" \
     -d "<snmp-info>
             <community>yRuSonoZ</community>
             <port>161</port>
             <retries>1</retries>
             <timeout>2000</timeout>
             <version>v2c</version>
         </snmp-info>" \
     -u admin:admin http://localhost:8980/opennms/rest/snmpConfig/10.1.1.1
```

Creates or updates a `<definition/>`-entry for IP address 10.1.1.1 in `snmp-config.xml`.

Example 2:

```
curl -v -X GET -u admin:admin http://localhost:8980/opennms/rest/snmpConfig/10.1.1.1
```

Returns the SNMP configuration for IP address 10.1.1.1 as defined in example 1.

API Version 2

Since Version 2 all attributes of a `<definition />` entry defined in `snmp-config.xsd` (http://xmlns.opennms.org/xsd/config/snmp) can be set or get via the interface - except it is only possible to set the configuration for one IP address and not for a range of IP addresses. This may change in the future.

The interface uses *SnmpInfo* objects for communication. Therefore it is possible to set for example v1 and v3 parameters in one request (e.g. `readCommunity` String and `privProtocol` String). However OpenNMS does not allow this. It is only allowed to set attributes which have no version restriction (e.g. timeout value) or the attributes which are limited to the version (e.g. `readCommunity` String if version is v1/v2c). The same is for getting data from the *API*, even if it is possible to store v1 and v3 parameters in one definition block in the `snmp-config.xml` manually, the *ReST API* will only return the parameters which match the version. If no version is defined, the default is assumed (both in *PUT* and *GET* requests).

The *SnmpInfo* schema is defined as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema
  elementFormDefault="qualified"
  version="1.0"
  targetNamespace="http://xmlns.opennms.org/xsd/config/snmp-info"
  xmlns:tns="http://xmlns.opennms.org/xsd/config/snmp-info"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="snmp-info" type="tns:snmpInfo"/>
  <xs:complexType name="snmpInfo">
    <xs:sequence>
      <xs:element name="authPassPhrase" type="xs:string" minOccurs="0"/>
      <xs:element name="authProtocol" type="xs:string" minOccurs="0"/>
      <xs:element name="community" type="xs:string" minOccurs="0"/>
      <xs:element name="contextEngineId" type="xs:string" minOccurs="0"/>
      <xs:element name="contextName" type="xs:string" minOccurs="0"/>
      <xs:element name="engineId" type="xs:string" minOccurs="0"/>
      <xs:element name="enterpriseId" type="xs:string" minOccurs="0"/>
      <xs:element name="maxRepetitions" type="xs:int" minOccurs="0"/>
      <xs:element name="maxRequestSize" type="xs:int" minOccurs="0"/>
      <xs:element name="maxVarsPerPdu" type="xs:int" minOccurs="0"/>
      <xs:element name="port" type="xs:int" minOccurs="0"/>
      <xs:element name="privPassPhrase" type="xs:string" minOccurs="0"/>
      <xs:element name="privProtocol" type="xs:string" minOccurs="0"/>
      <xs:element name="proxyHost" type="xs:string" minOccurs="0"/>
      <xs:element name="readCommunity" type="xs:string" minOccurs="0"/>
      <xs:element name="retries" type="xs:int" minOccurs="0"/>
      <xs:element name="securityLevel" type="xs:int" minOccurs="0"/>
      <xs:element name="securityName" type="xs:string" minOccurs="0"/>
      <xs:element name="timeout" type="xs:int" minOccurs="0"/>
      <xs:element name="version" type="xs:string" minOccurs="0"/>
      <xs:element name="writeCommunity" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

The following table shows all supported attributes, the mapping between `snmp-info.xsd` and `snmp-config.xsd`. It also shows the version limitations, default values and the restrictions - if any.

attribute snmp-info.xml	**attribute snmp-config.xml**
default	restricted to version
restriction	version
version	v1
-	"v1", "v2c" or "v3" are valid arguments. If an invalid or empty argument is provided "v1" is used.
port	port
161	-
Integer > 0	retries
retry	1
-	Integer > 0

attribute snmp-info.xml	attribute snmp-config.xml
timeout	timeout
3000	-
Integer > 0	maxVarsPerPdu
max-vars-per-pdu	10
-	Integer > 0
maxRepetitions	max-repetitions
2	-
Integer > 0	maxRequestSize
max-request-size	65535
-	Integer > 0
proxyHost	proxy-host
	-
	readCommunity
read-community	public
v1, v2c	
writeCommunity	write-community
private	v1, v2c
	securityName
security-name	opennmsUser
v3	
securityLevel	security-level
noAuthNoPriv	v3

attribute snmp-info.xml	attribute snmp-config.xml
Integer value, which can be null, 1, 2, or 3. 1 means noAuthNoPriv 2 means authNoPriv 3 means authPriv If you do not set the security level manually it is determined automatically: - if no authPassPhrase set the securityLevel is 1 - if a authPassPhrase and no privPassPhrase is set the security level is 2. - if a authPassPhrase and a privPassPhrase is set the security level is 3.	authPassPhrase
auth-passphrase	0p3nNMSv3

attribute snmp-info.xml	attribute snmp-config.xml
v3	
authProtocol	auth-protocol
MD5	v3
only MD5 or SHA are valid arguments	privPassPhrase
privacy-passphrase	0p3nNMSv3
v3	
privProtocol	privacy-protocol
DES	v3
only DES, AES, AES192 or AES256 are valid arguments.	engineId
engine-id	
v3	
contextEngineId	context-engine-id
	v3
	contextName
context-name	
v3	
enterpriseId	enterprise-id
	v3

Example 1:

```
curl -v -X PUT -H "Content-Type: application/xml" \
     -H "Accept: application/xml" \
     -d "<snmp-info>
             <readCommunity>yRuSonoZ</readCommunity>
             <port>161</port>
             <retries>1</retries>
             <timeout>2000</timeout>
             <version>v2c</version>
         </snmp-info>" \
     -u admin:admin http://localhost:8980/opennms/rest/snmpConfig/10.1.1.1
```

Creates or updates a `<definition/>`-entry for IP address 10.1.1.1 in `snmp-config.xml`.

Example 2:

```
curl -v -X GET -u admin:admin http://localhost:8980/opennms/rest/snmpConfig/10.1.1.1
```

Returns the SNMP configuration for IP address 10.1.1.1 as defined in example 1.

Example 3:

```
curl -v -X PUT -H "Content-Type: application/xml" \
     -H "Accept: application/xml" \
     -d "<snmp-info>
             <readCommunity>yRuSonoZ</readCommunity>
             <port>161</port>
             <retries>1</retries>
             <timeout>2000</timeout>
             <version>v1</version>
             <securityName>secret-stuff</securityName>
             <engineId>engineId</engineId>
         </snmp-info>" \
     -u admin:admin http://localhost:8980/opennms/rest/snmpConfig/10.1.1.1
```

Creates or updates a `<definition/>`-entry for IP address 10.1.1.1 in `snmp-config.xml` ignoring attributes `securityName` and `engineId`.

Example 4:

```
curl -v -X PUT -H "Content-Type: application/xml" \
     -H "Accept: application/xml" \
     -d "<snmp-info>
             <readCommunity>yRuSonoZ</readCommunity>
             <port>161</port>
             <retries>1</retries>
             <timeout>2000</timeout>
             <version>v3</version>
             <securityName>secret-stuff</securityName>
             <engineId>engineId</engineId>
         </snmp-info>" \
     -u admin:admin http://localhost:8980/opennms/rest/snmpConfig/10.1.1.1
```

Creates or updates a `<definition/>`-entry for IP address 10.1.1.1 in `snmp-config.xml` ignoring attribute `readCommunity`.

2.6.18. Users

Since users are not currently stored in the database, the ReST interface for them is not as full-fledged as that of nodes, etc.

IMPORTANT	You cannot use hibernate criteria for filtering. You may need to touch the `$OPENNMS_HOME/etc/users.xml` file on the filesystem for any addition or modification actions to take effect (see NMS-6469 for details).

GETs (Reading Data)

Parameter	Description
`/users`	Get a list of users.
`/users/{username}`	Get a specific user, by username.

POSTs (Adding Data)

Parameter	Description
`/users`	Add a user. If supplying a password it is assumed to be hashed or encrypted already, at least as of 1.12.5. To indicate that the supplied password uses the salted encryption algorithm rather than the older *MD5* based algorithm, you need to pass an element named `passwordSalt` with text `true` after the password element (or key/value pairs if using *JSON*).

PUTs (Modifying Data)

Parameter	Description
`/users/{username}`	Update an existing user's full-name, user-comments, password, passwordSalt and duty-schedule values.

2.7. ReST API Examples

2.7.1. Getting Graph data

While graphs aren't technically available via *ReST*, you can parse some *ReST* variables to get enough data to pull a graph. This isn't ideal because it requires multiple fetches, but depending on your use case, this may be adequate for you.

I'm in-lining some sample *PHP* code which should do this (not tested at all, cut & paste from old code I have that does not use the *ReST- interface, and/or coded straight into the browser so YMMV). If you go to your NMS and click the resource graphs, then right click the graph you want and hit _View Image* you will get the full *URL* that would need to be passed to pull that graph as a standalone image.

From that just take the *URL* and plug in the values you pulled from *ReST* to get a graph for whatever node you wanted.

```
function fetchit($thing, $user = "user", $pass = "pass") {
    $url = "http://localhost:8980/opennms";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url . $thing);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_USERAGENT, $useragent);
    curl_setopt($ch, CURLOPT_USERPWD, $user.':'.$pass);
    $data = curl_exec($ch);
    curl_close($ch);
    return $data;
}

// this assumes you already have found the nodeId via a previous REST call or some other means.  Provided
more as an example than what you might want.
function getNodeInterfaces($nodeId) {
    $data = fetchit("/rest/nodes/$nodeId/snmpinterfaces");
     return simplexml_load_string($data);
}

function fetchGraphs($nodeId) {
     $ints = getNodeInterfaces($nodeId);
     $chars = array('/','.',':','-',' ');
     $endtime = time();
     $starttime = (string)(time() - ($days * 24 * 60 * 60)) ;

     // use bcmath or a better version of PHP if  you don't want this hypocrisy here.
     $endtime = $endtime . '000';
     $starttime = $starttime . '000';

     for($i=0; $i<count($ints->snmpInterfaces); $i++) {
         $ifname = $ints->snmpInterfaces[$i]->snmpInterface->ifName;
         $mac = $ints->snmpInterfaces[$i]->snmpInterface->physAddr;
         $if = str_replace($chars, "_", $ifname);
         if ( strlen(trim($mac)) < 12 ) { $mac_and_if = $if; } else { $mac_and_if = $if .'-'. $mac; };

         $image = fetchit("$url/graph/graph.png?resource=node[$nodeId].interfaceSnmp[
$mac_and_if]&report=report=mib2.HCbits&start=$starttime&end=$endtime");
         // you can poop this to a file now, or set header('Content-type: image/png'); then print
"$image";
     }
}
```

2.7.2. provision.pl examples and notes

One way to test out the new *ReST* interface is to use `provision.pl`. If you run it you'll get a summary of the output, but it's not totally obvious how it all works.

Here is an example of adding a new node using the *ReST* interface:

```
# add a new foreign source called ubr
/usr/share/opennms/bin/provision.pl requisition add ubr
/usr/share/opennms/bin/provision.pl node add ubr 10341111 clownbox
/usr/share/opennms/bin/provision.pl node set ubr 10341111 city clownville
/usr/share/opennms/bin/provision.pl node set ubr 10341111 building clown-town-hall
/usr/share/opennms/bin/provision.pl node set ubr 10341111 parent-foreign-id 1122114
/usr/share/opennms/bin/provision.pl interface add ubr 10341111 10.1.3.4

# this is like a commit.  No changes will take effect until you import a foreign source
/usr/share/opennms/bin/provision.pl requisition import ubr
```

You will probably need to specify the username/password of an admin. To do this add:

```
--username=admin --password=clownnms
```

to the command line.

2.7.3. Debian (Lenny) Notes

For Lenny, you'll need to pull a package out of backports to make everything work right. Read http://backports.org/dokuwiki/doku.php?id=instructions for instructions on adding it to `sources.list`.

```
# install liburi-perl from backports
sudo apt-get -t lenny-backports install liburi-perl
```

2.7.4. Windows Powershell ReST

Example of using *Windows Powershell* to fill some asset fields with *ReST*.

```
# Installdate of Windows
$wmi = Get-WmiObject -Class Win32_OperatingSystem
$dateInstalled = $wmi.ConvertToDateTime($wmi.InstallDate)

# Serialnumber and manufacturer of server
Get-WmiObject win32_bios | select SerialNumber
$wmi = Get-WmiObject -Class win32_bios
$manufacturer = $wmi.Manufacturer

# Text file with a description of the server for the comments field
$comment = Get-Content "C:\Program Files\BGInfo\Info_Description.txt" | Out-String

$user ="admin"
$pass= "admin"

$secpasswd = ConvertTo-SecureString $user -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ($pass, $secpasswd)

$nodeid = Invoke-RestMethod -Uri
http://opennms.domain.nl:8980/opennms/rest/nodes?label=servername.domain.nl -Credential $cred
$nodeid = $nodeid.nodes.node.id

$uri="http://opennms.domain.nl:8980/opennms/rest/nodes/$nodeid/assetRecord"

Invoke-RestMethod -Uri
"http://opennms.massxess.nl:8980/opennms/rest/nodes/$nodeid/assetRecord/?building=133" -Credential $cred
-Method PUT
Invoke-RestMethod -Uri "$uri/?manufacturer=$manufacturer" -Credential $cred -Method PUT
Invoke-RestMethod -Uri "$uri/?dateInstalled=$dateInstalled" -Credential $cred -Method PUT
Invoke-RestMethod -Uri "$uri/?comment=$comment" -Credential $cred -Method PUT
```