

Developers Guide

Copyright (c) 2015 The OpenNMS Group, Inc.

OpenNMS 17.1.1

Last updated 2016-02-18 13:48:43 -05:00

Table of Contents

1. CORS Support	1
1.1. Why do I need CORS support?	1
1.2. How can I enable CORS support?	1
1.3. How can I configure CORS support?	1
2. ReST API	2
2.1. ReST URL	2
2.2. Authentication	2
2.3. Data format	2
2.4. Standard Parameters	2
2.5. Standard filter examples	3
2.6. Currently Implemented Interfaces	3
2.6.1. Acknowledgements	3
2.6.2. Alarm Statistics	4
2.6.3. Alarms	4
2.6.4. Events	5
2.6.5. Foreign Sources	6
2.6.6. Groups	7
2.6.7. Heatmap	8
2.6.8. KSC Reports	8
2.6.9. Maps	9
2.6.10. Measurements API	9
2.6.11. Nodes	12
2.6.12. Notifications	13
2.6.13. Outage Timelines	14
2.6.14. Outages	14
2.6.15. Requisitions	14
2.6.16. Realtime Console data	16
2.6.17. Scheduled Outages	16
2.6.18. SNMP Configuration	18
2.6.19. Users	26
2.6.20. SNMP Trap Northbounder Interface Configuration	26
2.6.21. Email Northbounder Interface Configuration	27
2.6.22. Javamail Configuration	28
2.6.23. Syslog Northbounder Interface Configuration	29
2.7. ReST API Examples	30
2.7.1. Getting Graph data	30
2.7.2. provision.pl examples and notes	31
2.7.3. Debian (Lenny) Notes	32
2.7.4. Windows Powershell ReST	32
3. Develop Documentation	34
3.1. File Structure in opennms-doc	34
3.2. Writing	34
3.2.1. Conventions for text formatting	34
3.2.2. Gotchas	35
3.3. Headings and document structure	35
3.4. Links	36
3.5. Admonitions and useful notes	37

3.6. Attributes	38
3.7. Comments	38
3.8. Tables	38
3.9. Include images	39
3.10. Code Snippets	41
3.10.1. Explicitly defined in the document	41
3.10.2. Included from an example file	42
3.10.3. Include parts of a file	42
3.11. Cheat Sheets and additional hints	43
4. AMQP Integration	44
4.1. Event Forwarder	44
4.1.1. Setup	44
4.1.2. Debugging	45
4.2. Event Receiver	45
4.2.1. Setup	45
4.2.2. Debugging	46
4.3. Alarm Northbounder	46
4.3.1. Setup	46
4.3.2. Debugging	46
4.4. Custom Processors	47

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 describes the RESTful interface for OpenNMS.

2.1. ReST URL

The base URL for Rest Calls is: <http://opennmserver: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
<code>limit</code>	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
<code>offset</code>	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 <i>URL</i> (for <i>GET</i>) or the form value (for <i>PUT</i> and <i>POST</i>). If so, the value will be used to add a filter to the result. By default, the operation is equality, unless the <code>comparator</code> 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:	
<code>eq</code>	Checks for equality
<code>ne</code>	Checks for non-equality
<code>ilike</code>	Case-insensitive wildcarding (% is the wildcard)
<code>like</code>	Case-sensitive wildcarding (% is the wildcard)
<code>gt</code>	Greater than
<code>lt</code>	Less than
<code>ge</code>	Greater than or equal
<code>le</code>	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). `nonnull` 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
<code>/events?eventUei=uei.opennms.org/internal/rtc/subscribe</code>	would return the first 10 events with the rtc subscribe UEI, (10 being the default limit for events)
<code>/events?eventUei=uei.opennms.org/internal/rtc/subscribe&limit=0</code>	would return all the rtc subscribe events (potentially quite a few)
<code>/events?id=100&comparator=gt</code>	would return the first 10 events with an id greater than 100
<code>/events?eventAckTime=nonnull</code>	would return the first 10 events that have a non-null Ack time (i.e. those that have been acknowledged)
<code>/events?eventAckTime=nonnull&id=100&comparator=gt&limit=20</code>	would return the first 20 events that have a non-null Ack time and an id greater than 100. Note that the <code>nonnull</code> value causes the comparator to be ignored for <code>eventAckTime</code>
<code>/events?eventAckTime=2008-07-28T04:41:30.530%2B12:00&id=100&comparator=gt&limit=20</code>	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. Also note that you must URL encode the plus sign when using GET.
<code>/events?orderBy=id&order=desc</code>	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
<code>/acks</code>	Get a list of acknowledgements.
<code>/acks/count</code>	Get the number of acknowledgements. (Returns plaintext, rather than XML or JSON.)
<code>/acks/{id}</code>	Get the acknowledgement specified by the given ID.

POSTs (Setting Data)

Resource	Description
<code>/acks</code>	Creates or modifies an acknowledgement for the given alarm <i>ID</i> or notification <i>ID</i> . To affect an alarm, set an <code>alarmId</code> parameter in the URL-encoded <i>POST</i> body; to affect a notification, set <code>notifyId</code> instead. An <code>action</code> parameter is also required, and may be one of <code>ack</code> , <code>unack</code> , <code>clear</code> , or <code>esc</code> (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
<code>/stats/alarms</code>	Returns statistics related to alarms. Accepts the same Hibernate parameters that you can pass to the <code>/alarms</code> ReST service.
<code>/stats/alarms/by-severity</code>	Returns the statistics related to alarms, one per severity. You can optionally pass a list of severities to the <code>severities</code> query parameter to limit it to the specified severities. (eg, <code>GET /opennms/rest/stats/alarms/by-severity?severities=MAJOR,CRITICAL</code>).

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
<code>/alarms</code>	Get a list of alarms.
<code>/alarms/count</code>	Get the number of alarms. (Returns plaintext, rather than <i>XML</i> or <i>JSON</i> .)
<code>/alarms/{id}</code>	Get the alarms specified by the given <i>ID</i> .

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

Resource	Description
<code>/alarms?comparator=ge&severity=MINOR</code>	Get the alarms with a severity greater than or equal to <i>MINOR</i> .

PUTs (Modifying Data)

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

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

New in OpenNMS 1.11.0

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

Resource	Description
<code>/alarms/{id}?clear=true</code>	Clears an alarm.
<code>/alarms/{id}?escalate=true</code>	Escalates an alarm. eg, NORMAL MINOR, MAJOR CRITICAL, etc.
<code>/alarms?x=y& &clear=true</code>	Clears alarms matching the additional query parameters.
<code>/alarms?x=y& &escalate=true</code>	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
<code>/events</code>	Get a list of events. The default for offset is 0, and the default for limit is 10. To get all results, use <code>limit=0</code> as a parameter on the URL (ie, <code>GET /events?limit=0</code>).
<code>/events/count</code>	Get the number of events. (Returns plaintext, rather than <code>XML</code> or <code>JSON</code> .)
<code>/events/{id}</code>	Get the event specified by the given <code>ID</code> .

PUTs (Modifying Data)

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

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

POSTs (Adding Data)

POST requires XML (application/xml) or JSON (application/json) as its Content-Type.

TIP See `/${OPENNMS_HOME}/share/xsds/event.xsd` for the reference schema.

Resource	Description
<code>/events</code>	Publish an event on the event bus.

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
<code>/foreignSources</code>	Get all active foreign sources.
<code>/foreignSources/default</code>	Get the active default foreign source.
<code>/foreignSources/deployed</code>	Get the list of all deployed (active) foreign sources.
<code>/foreignSources/deployed/count</code>	Get the number of deployed foreign sources. (Returns plaintext, rather than XML or JSON.)
<code>/foreignSources/{name}</code>	Get the active foreign source named {name}.
<code>/foreignSources/{name}/detectors</code>	Get the configured detectors for the foreign source named {name}.
<code>/foreignSources/{name}/detectors/{detector}</code>	Get the specified detector for the foreign source named {name}.
<code>/foreignSources/{name}/policies</code>	Get the configured policies for the foreign source named {name}.
<code>/foreignSources/{name}/policies/{policy}</code>	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
<code>/foreignSources</code>	Add a foreign source.
<code>/foreignSources/{name}/detectors</code>	Add a detector to the named foreign source.
<code>/foreignSources/{name}/policies</code>	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
<code>/foreignSources/{name}</code>	Modify a foreign source with the given name.

DELETEs (Removing Data)

Resource	Description
<code>/foreignSources/{name}</code>	Delete the named foreign source.
<code>/foreignSources/{name}/detectors/{detector}</code>	Delete the specified detector from the named foreign source.
<code>/foreignSources/{name}/policies/{policy}</code>	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
<code>/groups</code>	Get a list of groups.
<code>/groups/{groupname}</code>	Get a specific group, given a group name.
<code>/groups/{groupname}/users</code>	Get the users for a group, given a group name. (new in OpenNMS 14)
<code>/groups/{groupname}/categories</code>	Get the categories associated with a group, given a group name. (new in OpenNMS 14)

POSTs (Adding Data)

Resource	Description
<code>/groups</code>	Add a new group.

PUTs (Modifying Data)

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

DELETES (Removing Data)

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

2.6.7. Heatmap

GETs (Reading Data)

Resource	Description
<code>/heatmap/outages/categories</code>	Sizes and color codes based on outages for nodes grouped by <i>Surveillance Categories</i>
<code>/heatmap/outages/foreignSources</code>	Sizes and color codes based on outages for nodes grouped by <i>Foreign Source</i>
<code>/heatmap/outages/monitoredServices</code>	Sizes and color codes based on outages for nodes grouped by monitored services
<code>/heatmap/outages/nodesByCategory/{category}</code>	Sizes and color codes based on outages for nodes associated with a specific <i>Surveillance Category</i>
<code>/heatmap/outages/nodesByForeignSource/{foreignSource}</code>	Sizes and color codes based on outages for nodes associated with a specific <i>Foreign Source</i>
<code>/heatmap/outages/nodesByMonitoredService/{monitoredService}</code>	Sizes and color codes based on outages for nodes providing a specific monitored service

Resource	Description
<code>/heatmap/alarms/categories</code>	Sizes and color codes based on alarms for nodes grouped by <i>Surveillance Categories</i>
<code>/heatmap/alarms/foreignSources</code>	Sizes and color codes based on alarms for nodes grouped by <i>Foreign Source</i>
<code>/heatmap/alarms/monitoredServices</code>	Sizes and color codes based on alarms for nodes grouped by monitored services
<code>/heatmap/alarms/nodesByCategory/{category}</code>	Sizes and color codes based on alarms for nodes associated with a specific <i>Surveillance Category</i>
<code>/heatmap/alarms/nodesByForeignSource/{foreignSource}</code>	Sizes and color codes based on alarms for nodes associated with a specific <i>Foreign Source</i>
<code>/heatmap/alarms/nodesByMonitoredService/{monitoredService}</code>	Sizes and color codes based on alarms for nodes providing a specific monitored service

2.6.8. KSC Reports

GETs (Reading Data)

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

PUTs (Modifying Data)

Resource	Description
<code>/ksc/{reportId}</code>	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](#)

2.6.9. Maps

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

GETs (Reading Data)

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

POSTs (Adding Data)

Resource	Description
<code>/maps</code>	Add a map.

PUTs (Modifying Data)

Resource	Description
<code>/maps/{id}</code>	Update the properties of the map with the given <i>ID</i> .

DELETEs (Removing Data)

Resource	Description
<code>/maps/{id}</code>	Delete the map with the given <i>ID</i> .

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
<code>/measurements/{resourceId}/{attribute}</code>	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.
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.
interval	null	Duration in milliseconds, used by strategies that implement late aggregation.
heartbeat	null	Duration in milliseconds, used by strategies that implement late aggregation.
aggregation	AVERAGE	Consolidation function used. Can typically be AVERAGE , MIN or MAX . Depends on RRA definitions.
fallback-attribute		Secondary attribute that will be queried in the case the primary attribute does not exist.

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 below.

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].interfaceSntp[eth0-04013f75f101]",
      "transient": "false"
    },
    {
      "aggregation": "AVERAGE",
      "attribute": "ifHCOutOctets",
      "label": "ifHCOutOctets",
      "resourceId": "nodeSource[NODES:1424038123222].interfaceSntp[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
<code>/nodes</code>	Get a list of nodes. This includes the ID and node label.
<code>/nodes/{id}</code>	Get a specific node, by ID.
<code>/nodes/{id}/ipinterfaces</code>	Get the list of IP interfaces associated with the given node.
<code>/nodes/{id}/ipinterfaces/{ipAddress}</code>	Get the IP interface for the given node and IP address.
<code>/nodes/{id}/ipinterfaces/{ipAddress}/services</code>	Get the list of services associated with the given node and IP interface.
<code>/nodes/{id}/ipinterfaces/{ipAddress}/services/{service}</code>	Get the requested service associated with the given node, IP interface, and service name.

GETs (Reading Data)

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

POSTs (Setting Data)

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

PUTs (Modifying Data)

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

DELETEs (Removing Data)

Parameter	Description
<code>/sched-outages/{outageName}</code>	to delete a specific outage.
<code>/sched-outages/{outageName}/collectd/{package}</code>	to remove a specific outage from a collectd's package.

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
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

attribute snmp-info.xml	attribute snmp-config.xml
	securityName
security-name	opennmsUser
v3	
securityLevel	security-level
noAuthNoPriv	v3

attribute snmp-info.xml	attribute snmp-config.xml
<p>Integer value, which can be null, 1, 2, or 3.</p> <ul style="list-style-type: none"> 1 means noAuthNoPriv2 means authNoPriv3 means authPriv <p>If you do not set the security level manually it is determined automatically:</p> <ul style="list-style-type: none"> if no authPassPhrase set the securityLevel is 1if 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. 	<p>authPassPhrase</p>
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

Resource	Description
<code>/config/email-nbi/destinations</code>	Adds a new or overrides an existing destination.

PUTs (Modifying Data)

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

On a successful request, the Email NBI will be notified about the configuration change.

Resource	Description
<code>/config/email-nbi/destinations/{name}</code>	Updates the content of the destination named {name}

DELETES (Remove Data)

On a successful request, the Email NBI will be notified about the configuration change.

Resource	Description
<code>/config/email-nbi/destinations/{name}</code>	Updates the content of the destination named {name}

2.6.22. Javamail Configuration

GETs (Reading Data)

Resource	Description
<code>/config/javamail/default/readmail</code>	Get the name of the default readmail config.
<code>/config/javamail/default/sendmail</code>	Get the name of the default sendmail config.
<code>/config/javamail/readmails</code>	Get the name of all the existing readmail configurations.
<code>/config/javamail/sendmails</code>	Get the name of all the existing sendmail configurations.
<code>/config/javamail/end2ends</code>	Get the name of all the existing end2end mail configurations.
<code>/config/javamail/readmails/{name}</code>	Get the content of the readmail configuration named {name}
<code>/config/javamail/sendmails/{name}</code>	Get the content of the sendmail configuration named {name}
<code>/config/javamail/end2ends/{name}</code>	Get the content of the end2end mail configuration named {name}

POSTs (Adding/Updating Data)

POST requires form data using `application/xml` or `application/json` as a Content-Type.

On a successful request, the Email NBI will be notified about the configuration change.

Resource	Description
<code>/config/javamail/readmails</code>	Adds a new or overrides an existing readmail configuration.
<code>/config/javamail/sendmails</code>	Adds a new or overrides an existing sendmail configuration.
<code>/config/javamail/end2ends</code>	Adds a new or overrides an existing end2ends mail configuration.


```

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=$endtime");
        // you can pop this to a file now, or set header('Content-type: image/png'); then print
        "$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=clovnms
```

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

```


fail.

To start a document:

```
[[unique-id-verbose-is-ok]]  
= The Document Title
```

If you are including the document inside another document and you need to push the headings down to the right level in the output, the `leveloffset` attribute is used.

Subsequent headings in a document should use the following syntax:

```
== Subheading  
... content here ...  
=== Subsubheading  
content here ...
```

3.4. Links

When you need to link to other parts of the manual you use the target id. To use a target id you follow this syntax:

```
<<doc-guidelines-links>>
```

This will render as: [\[doc-guidelines-links\]](#)

NOTE | To use the target id in you document simply write the target id in your text, for example:

see <<target-id>>

this should suffice for most cases.

If you need to link to another document with your own link text, then follow this procedure:

```
<<target-id, link text that fits in the context>>
```

NOTE | Having lots of linked text may work well in a web context but is a distracting in print. The documentation we are creating is intended for both mediums so be considerate of this in your usage.

If you wish to use an external link, they are are added as:

```
http://www.opennms.org/[Link text here]
```

This will render in the output as: [Link text here](#)

For short links it may be beneficial not to use accompanying link text:


```
TIP: Tiptext. +  
Line 2.
```

Which is rendered as:

```
TIP | Tiptext.  
      | Line 2.
```

```
NOTE | Remember to write these in full caps. There is no easy manner in which to add new admonitions, do not  
      | create your own.
```

3.6. Attributes

Common attributes you can use in documents:

- `{opennms-version}` - rendered as "17.1.1"

These can substitute part of URLs that point to, for example, API docs or source code. Note that `opennms-git-tag` also handles the case of `snapshot/master`.

Sample AsciiDoc attributes which can be used:

- `{docdir}` - root directory of the documents
- `{nbsp}` - non-breaking space

3.7. Comments

There's a separate build that includes comments. When the comments are used they show up with a yellow background. This build doesn't run by default, but after a normal build, you can use `make annotated` to create a build yourself. You can use the resulting 'annotated' page to search for content as the full manual is a single page.

To write a comment:

```
// this is a comment
```

Comments are not visible in the standard build. Comment blocks won't be included in the output of any build. The syntax for a comment block is:

```
////  
Note that includes in here will still be processed, but not make it into the output.  
That is, missing includes here will still break the build!  
////
```

3.8. Tables

For representing structured information you can use tables. A table is constructed in the following manner:

Include this tagged part in the documentation using the tag parameter

```
[source,xml]
----
include::../configs/poller-configuration.xml[tags=IcmpServiceConfig]
----
```

This is how it rendered

```
<service name="ICMP" interval="300000" user-defined="false" status="on">
  <parameter key="retry" value="2" />
  <parameter key="timeout" value="3000" />
  <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response" />
  <parameter key="rrd-base-name" value="icmp" />
  <parameter key="ds-name" value="icmp" />
</service>
```

NOTE | Spaces and tabs are taken from the original file.

3.11. Cheat Sheets and additional hints

For instructions on how to build your own version of the manual:

- [readme](#)

The documentation uses the AsciiDoc format. There are a number of guides that will help you to get started with using AsciiDoc:

- [Aciidoc Reference](#)
- [AsciiDoc FAQ](#)
- [AsciiDoc cheatsheet](#)
- [AsciiDoc Cheatsheet](#)

For other resources, to gain familiarity with AsciiDoc, you can visit:

- [AsciiDoc User Manual](#)
- [AsciiDoc Maven Plugin](#)
- [AsciiDoc discussion list](#)
- [AsciiDoc issue tracker](#)
- [Docbook to AsciiDoc](#)
- [How to create handsome PDF documents without frustration](#)


```
camel:route-info forwardAlarm
```

4.4. Custom Processors

If your integration requires specific filtering and or formatting, you can write your own processor by implementing the `org.apache.camel.Processor` interface.

For example, we can implement a custom processor used for event forwarding:

```
import org.apache.camel.Exchange;
import org.apache.camel.Processor;

public class MyEventProcessor implements Processor {
    @Override
    public void process(final Exchange exchange) throws Exception {
        final Event event = exchange.getIn().getBody(Event.class);

        // Filtering
        if (!shouldForward(event)) {
            exchange.setProperty(Exchange.ROUTE_STOP, Boolean.TRUE);
            return;
        }

        // Transforming
        MyDTO eventAsDTO = toDTO(event);
        exchange.getIn().setBody(eventAsDTO, MyDTO.class);
    }
}
```

In order to use the processor, package it as a bundle, and expose it to the OSGi service registry using:

```
<bean id="myEventProcessor" class="org.opennms.integrations.evilcorp.MyEventProcessor" />
<service id="myEventProcessorService" ref="myEventProcessor" interface="org.apache.camel.Processor">
  <service-properties>
    <entry key="name" value="evilcorp-event-forwarder-processor"/>
  </service-properties>
</service>
```

Once your bundle is in the Karaf container, you can update the loaded you can refer to your processor with:

```
config:edit org.opennms.features.amqp.eventforwarder
propset processorName evilcorp-event-forwarder-processor
config:update
```

If the event forwarder feature was already started, it should automatically restart and start using the new processor. Otherwise, you can start the feature with:

```
feature:install opennms-amqp-event-forwarder
```