

Developers Guide

Copyright (c) 2015-2021 The OpenNMS Group, Inc.

OpenNMS Meridian 2020.1.17, Last updated 2021-12-16 01:57:36 UTC

Table of Contents

1. Setup a development system	1
1.1. Operating System/Environment	1
1.2. Installation	1
1.3. Tooling	3
1.4. Useful links	3
1.4.1. General	3
1.4.2. Installation/Setup	3
2. Topology	4
2.1. Info Panel Items	4
2.1.1. Programmatic	4
2.1.2. Scriptable	5
2.2. GraphML	9
2.2.1. Create/Update/Delete GraphML Topology	10
2.2.2. Supported Attributes	11
2.2.3. Focus Strategies	12
2.2.4. Icons	13
2.2.5. Vertex Status Provider	13
2.2.6. Edge Status Provider	13
2.2.7. Layers	14
2.2.8. Breadcrumbs	17
2.3. Topologies Updates	20
2.3.1. OnmsTopologyUpdater	20
2.3.2. OnmsTopologyRef	21
2.3.3. OnmsTopologyMessage	21
2.3.4. OnmsTopologyProtocol	21
2.3.5. TopologyMessageStatus	21
2.3.6. OnmsTopology	22
2.3.7. OnmsTopologyVertex	22
2.3.8. OnmsTopologyEdge	22
2.3.9. OnmsTopologyPort	22
2.3.10. OnmsTopologyConsumer	22
3. Graph Service API	23
3.1. Terminology	23
3.2. Graph Model	23
3.2.1. Generic Graph Model	23
3.2.2. Domain Graph Model	24
3.2.3. Immutability	24
3.3. Providers	24

3.3.1. Meta Information	24
3.4. Cache	25
3.5. Search	25
3.6. Focus / Semantic Zoom Level / View	25
3.7. Listening for Changes	26
3.8. Persistence	26
3.9. Enrichment	26
3.9.1. Build in Enrichment.....	26
3.10. ReST API	26
3.11. Limitations.....	27
4. CORS Support	28
4.1. Why do I need CORS support?.....	28
4.2. How can I enable CORS support?.....	28
4.3. How can I configure CORS support?	28
5. ReST API	29
5.1. ReST URL.....	29
5.2. Authentication.....	29
5.3. Data format	29
5.4. Standard Parameters.....	29
5.5. Standard filter examples	30
5.6. HTTP Return Codes	31
5.7. Identifying Resources	31
5.8. Expose ReST services via OSGi	32
5.8.1. Define a ReST interface	32
5.8.2. Implement a ReST interface.....	33
5.8.3. Export the ReST service.....	33
5.9. Currently Implemented Interfaces.....	34
5.9.1. Acknowledgements	34
5.9.2. Alarm Statistics	35
5.9.3. Alarms	35
5.9.4. Alarm History	37
5.9.5. Events	37
5.9.6. Categories	38
5.9.7. Flow API	39
5.9.8. Flow Classification API	47
5.9.9. Foreign Sources.....	53
5.9.10. Graph API	55
5.9.11. Groups.....	66
5.9.12. Heatmap	67
5.9.13. Monitored Services.....	68
5.9.14. IP Interfaces	68

5.9.15. KSC Reports	69
5.9.16. Maps.....	70
5.9.17. Measurements API.....	70
5.9.18. Meta-Data	77
5.9.19. Nodes.....	79
5.9.20. Notifications.....	80
5.9.21. Outage Timelines	81
5.9.22. Outages	81
5.9.23. Requisitions	81
5.9.24. Resources API	84
5.9.25. Realtime Console data	87
5.9.26. Scheduled Outages	88
5.9.27. SNMP Configuration.....	90
5.9.28. SNMP Interfaces	99
5.9.29. Users	100
5.9.30. User Defined Links.....	101
5.9.31. SNMP Trap Northbunder Interface Configuration.....	102
5.9.32. Email Northbunder Interface Configuration.....	103
5.9.33. Javamail Configuration.....	104
5.9.34. Syslog Northbunder Interface Configuration	106
5.9.35. Business Service Monitoring.....	107
5.9.36. Discovery	110
5.9.37. Situation Feedback.....	112
5.10. ReST API Examples	114
5.10.1. Getting Graph data.....	114
5.10.2. provision.pl examples and notes	115
5.10.3. Debian (Lenny) Notes	116
5.10.4. Windows Powershell ReST.....	116
6. Develop Documentation.....	118
6.1. File Structure in opennms-doc	118
6.2. Writing	118
6.2.1. Conventions for text formatting	119
6.2.2. Gotchas	119
6.3. Headings and document structure.....	120
6.4. Links	120
6.5. Admonitions and useful notes.....	121
6.6. Attributes	122
6.7. Comments.....	123
6.8. Tables	123
6.9. Include images	124
6.10. Code Snippets	126

6.10.1. Explicitly defined in the document	126
6.10.2. Included from an example file	127
6.10.3. Include parts of a file.....	127
6.11. Cheat sheets and additional hints	128
6.12. Migrating content from project wiki	129
7. AMQP Integration	131
7.1. Event Forwarder.....	131
7.1.1. Setup	132
7.1.2. Debugging.....	132
7.2. Event Receiver	132
7.2.1. Setup	133
7.2.2. Debugging.....	133
7.3. Alarm Northbounder.....	133
7.3.1. Setup	134
7.3.2. Debugging.....	134
7.4. Custom Processors	134
8. Design and Styleguidelines	136
8.1. Jasper Report Guideline	136

Chapter 1. Setup a development system

This guide describes the requirements and how to get started with the development of the OpenNMS project.

1.1. Operating System/Environment

To build/compile OpenNMS requires running a *nix system. You do not need to run it physically, a virtual machine is sufficient. We recommend one of the following:

- [Linux Mint](#) with Cinnamon Desktop environment
- [Ubuntu Desktop](#)
- Mac OS X



This documentation describes a Debian-based desktop environment.

1.2. Installation

Follow these instructions to set up your environment. They may vary depending on your operating system:

```

# add OpenNMS as repository to install icmp and such
echo "deb http://debian.opennms.org stable main" >
/etc/apt/sources.list.d/opennms.list
echo "deb-src http://debian.opennms.org stable main" >>
/etc/apt/sources.list.d/opennms.list
# Add pgp key
wget -O - https://debian.opennms.org/OPENNMS-GPG-KEY | apt-key add -

# overall update
apt-get update

# install stuff
apt-get install -y software-properties-common
apt-get install -y git-core
apt-get install -y nsis

# install Oracle Java 8 JDK
# this setup is based on: http://www.webupd8.org/2014/03/how-to-install-oracle-java-8-
in-debian.html
add-apt-repository -y ppa:webupd8team/java
apt-get update
apt-get install -y oracle-java8-installer
apt-get install -y oracle-java8-set-default

# install and configure PostgreSQL
apt-get install -y postgresql
echo "local  all                postgres peer" >
/etc/postgresql/9.3/main/pg_hba.conf
echo "local  all                all      peer" >>
/etc/postgresql/9.3/main/pg_hba.conf
echo "host   all                all      127.0.0.1/32 trust" >>
/etc/postgresql/9.3/main/pg_hba.conf
echo "host   all                all      ::1/128  trust" >>
/etc/postgresql/9.3/main/pg_hba.conf
# restart postgres to use new configs
/etc/init.d/postgresql restart

# install OpenNMS basic dependencies
apt-get install -y maven
apt-get install -y jicmp jicmp6
apt-get install -y jrrd

# clone opennms
mkdir -p ~/dev/opennms
git clone https://github.com/OpenNMS/opennms.git ~/dev/opennms

```

After this you should be able to build OpenNMS:

```
cd ~/dev/opennms
./clean.pl
./compile.pl -DskipTests
./assemble.pl -p dir
```

For more information on how to build OpenNMS from source, check this wiki [Install from Source](#).

After successfully building OpenNMS, please follow the wiki [Running OpenNMS](#).

1.3. Tooling

We recommend the following toolset:

- IDE: [IntelliJ IDEA Ultimate](#)
- DB-Tool: [DBBeaver](#) or [Postgres Admin - pgAdmin](#)
- Graphing: [yEd](#)
- Other: [atom.io](#)

1.4. Useful links

1.4.1. General

- [source code](#) hosted on GitHub
- [Our Wiki](#), especially the start page. It points you in the right direction.
- Our [issue/bug tracker](#)
- [Vagrant box](#) to set up a virtual box to build OpenNMS
- [vagrant box](#) to set up a virtual box to run OpenNMS

1.4.2. Installation/Setup

- <http://www.opennms.eu/docs/opennms-community-welcome-guide/0.0.5-SNAPSHOT/>
- <http://www.opennms.org/wiki/Installation:Source>
- http://www.opennms.org/wiki/Developing_with_Git
- http://www.opennms.org/wiki/Eclipse_and_OpenNMS
- http://www.opennms.org/wiki/IDEA_and_OpenNMS

Chapter 2. Topology

2.1. Info Panel Items



This section is under development. All provided examples or code snippet may not fully work. However they are conceptionally correct and should point in the right direction.

Each element in the *Info Panel* is defined by an `InfoPanelItem` object.

All available `InfoPanelItem` objects are sorted by the order. This allows to arrange the items in a custom order. After the elements are ordered, they are put below the *SearchBox* and the *Vertices in Focus* list.

2.1.1. Programmatic

It is possible to add items to the *Info Panel* in the *Topology UI* by simply implementing the interface `InfoPanelItemProvider` and expose its implementation via OSGi.

Simple Java InfoPanelItemProvider

```
public class ExampleInfoPanelItemProvider implements InfoPanelItemProvider {
    @Override
    public Collection<? extends InfoPanelItem> getContributions(GraphContainer
container) {
        return Collections.singleton(
            new DefaultInfoPanelItem() ①
                .withTitle("Static information") ②
                .withOrder(0) ③
                .withComponent(
                    new com.vaadin.v7.ui.Label("I am a static component") ④
                )
        );
    }
}
```

- ① The default implementation of `InfoPanelItem`. You may use `InfoPanelItem` instead if the default implementation is not sufficient.
- ② The title of the `InfoPanelItem`. It is shown above the component.
- ③ The order.
- ④ A Vaadin component which actually describes the custom component.

In order to show information based on a selected vertex or edge, one must inherit the classes `EdgeInfoPanelItemProvider` or `VertexInfoPanelItemProvider`. The following example shows a custom `EdgeInfoPanelItemProvider`.

```
public class ExampleEdgeInfoPanelItemProvider extends EdgeInfoPanelItemProvider {
    @Override
    protected boolean contributeTo(EdgeRef ref, GraphContainer graphContainer) { ①
        return "custom-namespace".equals(ref.getNamespace()); // only show if of
        certain namespace
    }

    @Override
    protected InfoPanelItem createInfoPanelItem(EdgeRef ref, GraphContainer
    graphContainer) { ②
        return new DefaultInfoPanelItem()
            .withTitle(ref.getLabel() + " Info")
            .withOrder(0)
            .withComponent(
                new com.vaadin.v7.ui.Label("Id: " + ref.getId() + ",
                Namespace: " + ref.getNamespace())
            );
    }
}
```

- ① Is invoked if one and only one edge is selected. It determines if the current edge should provide the *InfoPanelItem* created by *createInfoPanelItem*.
- ② Is invoked if one and only one edge is selected. It creates the *InfoPanelItem* to show for the selected edge.

Implementing the provided interfaces/classes, is not enough to have it show up. It must also be exposed via a `blueprint.xml` to the OSGi service registry. The following `blueprint.xml` snippet describes how to expose any custom *InfoPanelItemProvider* implementation to the OSGi service registry and have the *Topology UI* pick it up.

blueprint.xml snippet

```
<service interface="org.opennms.features.topology.api.info.InfoPanelItemProvider"> ①
    <bean class="ExampleInfoPanelItemProvider" /> ②
</service>
```

- ① The service definition must always point to *InfoPanelItemProvider*.
- ② The bean implementing the defined interface.

2.1.2. Scriptable

By simply dropping JinJava templates (with file extension `.html`) to `$OPENNMS_HOME/etc/infopanel` a more scriptable approach is available. For more information on JinJava refer to <https://github.com/HubSpot/jinjava>.

The following example describes a very simple JinJava template which is always visible.

Static scriptable template

```
{% set visible = true %} ①  
{% set title = "Static information" %} ②  
{% set order = -700 %} ③  
  
This information is always visible ④
```

- ① Makes this always visible
- ② Defines the title
- ③ Each info panel item is ordered at the end. Making it -700 makes it very likely to pin this to the top of the info panel item.

A template showing custom information may look as following:

Vertex specific template

```
{% set visible = vertex != null && vertex.namespace == "custom" &&  
vertex.customProperty is defined %} ①  
{% set title = "Custom Information" %}  
  
<table width="100%" border="0">  
  <tr>  
    <td colspan="3">This information is only visible if a vertex with namespace  
"custom" is selected</td>  
  </tr>  
  <tr>  
    <td align="right" width="80">Custom Property</td>  
    <td width="14"></td>  
    <td align="left">{{ vertex.customProperty }}</td>  
  </tr>  
</table>
```

- ① This template is only shown if a vertex is selected and the selected namespace is "custom".

It is also possible to show performance data.

Including resource graphs

One can include resource graphs into the info panel by using the following HTML element:

```
<div class="graph-container" data-resource-id="RESOURCE_ID" data-graph-name=  
"GRAPH_NAME"></div>
```

Optional attributes `data-graph-start` and `data-graph-end` can be used to specify the displayed time range in seconds since epoch.

Measurements API template (memory usage)

```

{# Example template for a simple memory statistic provided by the netsnmp agent #}
{% set visible = node != null && node.sysObjectId == ".1.3.6.1.4.1.8072.3.2.10" %}
{% set order = 110 %}

{# Setting the title #}
{% set title = "System Memory" %}

{# Define resource Id to be used #}
{% set resourceId = "node[" + node.id + "].nodeSnmplib[" %}

{# Define attribute Id to be used #}
{% set attributeId = "hrSystemUptime" %}

{% set total = measurements.getLastValue(resourceId, "memTotalReal")/1000/1024 %}
{% set avail = measurements.getLastValue(resourceId, "memAvailReal")/1000/1024 %}

<table border="0" width="100%">
  <tr>
    <td width="80" align="right" valign="top">Total</td>
    <td width="14"></td>
    <td align="left" valign="top" colspan="2">
      {{ total|round(2) }} GB(s)
    </td>
  </tr>
  <tr>
    <td width="80" align="right" valign="top">Used</td>
    <td width="14"></td>
    <td align="left" valign="top" colspan="2">
      {{ (total-avail)|round(2) }} GB(s)
    </td>
  </tr>
  <tr>
    <td width="80" align="right" valign="top">Available</td>
    <td width="14"></td>
    <td align="left" valign="top" colspan="2">
      {{ avail|round(2) }} GB(s)
    </td>
  </tr>
  <tr>
    <td width="80" align="right" valign="top">Usage</td>
    <td width="14"></td>
    <td align="left" valign="top">
      <meter style="width:100% min="0" max="{{ total }}" low="{{ 0.5*total }}"
high="{{ 0.8*total }}" value="{{ total-avail }}" optimum="0"/>
    </td>
    <td width="1">
      &nbsp;{{ ((total-avail)/total*100)|round(2) }}%
    </td>
  </tr>
</table>

```

Measurements API template (uptime)

```
{# Example template for the system uptime provided by the netsnmp agent #}
{% set visible = node != null && node.sysObjectId == ".1.3.6.1.4.1.8072.3.2.10" %}
{% set order = 100 %}

{# Setting the title #}
{% set title = "System Uptime" %}

{# Define resource Id to be used #}
{% set resourceId = "node[" + node.id + "].nodeSnmp[]" %}

{# Define attribute Id to be used #}
{% set attributeId = "hrSystemUptime" %}

<table border="0" width="100%">
  <tr>
    <td width="80" align="right" valign="top">getLastValue()</td>
    <td width="14"></td>
    <td align="left" valign="top">
      {# Querying the last value via the getLastValue() method: #}

      {% set last = measurements.getLastValue(resourceId,
attributeId)/100.0/60.0/60.0/24.0 %}
      {{ last|round(2) }} day(s)
    </td>
  </tr>
  <tr>
    <td width="80" align="right" valign="top">query()</td>
    <td width="14"></td>
    <td align="left" valign="top">
      {# Querying the last value via the query() method. A custom function
'currentTimeMillis()' in
      the namespace 'System' is used to get the timestamps for the query: #}

      {% set end = System:currentTimeMillis() %}
      {% set start = end - (15 * 60 * 1000) %}

      {% set values = measurements.query(resourceId, attributeId, start, end,
300000, "AVERAGE") %}

      {# Iterating over the values in reverse order and grab the first value
which is not NaN #}
      {% set last = "NaN" %}
      {% for value in values|reverse %}
        {%- if value != "NaN" && last == "NaN" %}
          {{ (value/100.0/60.0/60.0/24.0)|round(2) }} day(s)
          {% set last = value %}
        {% endif %}
      {%- endfor %}
    </td>
  </tr>
</table>
```

```

</tr>
<tr>
  <td width="80" align="right" valign="top">Graph</td>
  <td width="14"></td>
  <td align="left" valign="top">
    {# We use the start and end variable here to construct the graph's Url: #}

    
  </td>
</tr>
</table>

```

2.2. GraphML

In OpenNMS Meridian the `GraphMLTopologyProvider` uses `GraphML` formatted files to visualize graphs.

GraphML is a comprehensive and easy-to-use file format for graphs. It consists of a language core to describe the structural properties of a graph and a flexible extension mechanism to add application-specific data. [...] Unlike many other file formats for graphs, GraphML does not use a custom syntax. Instead, it is based on XML and hence ideally suited as a common denominator for all kinds of services generating, archiving, or processing graphs.

— <http://graphml.graphdrawing.org>

OpenNMS Meridian does not support the full feature set of GraphML. The following features are not supported: Nested graphs, Hyperedges, Ports and Extensions. For more information about GraphML refer to the [Official Documentation](#).

A basic graph definition using GraphML usually consists of the following GraphML elements:

- Graph element to describe the graph
- Key elements to define custom properties, each element in the GraphML document can define as data elements
- Node and Edge elements
- Data elements to define custom properties, which OpenNMS Meridian will then interpret.

A very minimalistic example is given below:

```

<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
    http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <!-- key section -->
  <key id="label" for="all" attr.name="label" attr.type="string"></key>
  <key id="namespace" for="graph" attr.name="namespace" attr.type="string"></key>

  <!-- shows up in the menu -->
  <data key="label">Minimalistic GraphML Topology Provider</data> ①
  <graph id="minicmalistic"> ②
    <data key="namespace">minimalistic</data> ③
    <node id="node1"/> ④
    <node id="node2"/>
    <node id="node3"/>
    <node id="node4"/>
  </graph>
</graphml>

```

- ① The optional label of the menu entry.
- ② The graph definition.
- ③ Each graph must have a namespace, otherwise OpenNMS Meridian refuses to load the graph.
- ④ Node definitions.

2.2.1. Create/Update/Delete GraphML Topology

In order to create a GraphML Topology, a valid GraphML xml file must exist. Afterwards this is send to the OpenNMS Meridian REST API to create it:

```

curl -X POST -H "Content-Type: application/xml" -u admin:admin -d@graph.xml
'http://localhost:8980/opennms/rest/graphml/topology-name'

```

The `topology-name` is a unique identifier for the Topology. If a `label` property is defined for the Graphml element this is used to be displayed in the Topology UI, otherwise the `topology-name` defined here is used as a fallback.

To delete an already existing Topology a HTTP DELETE request must be send:

```

curl -X DELETE -u admin:admin 'http://localhost:8980/opennms/rest/graphml/topology-
name'

```

There is no PUT method available. In order to update an existing GraphML Topology one must first delete and afterwards re-create it.



Even if the HTTP Request was successful, it does not mean, that the Topology is actually loaded properly. The HTTP Request states that the Graph was successfully received, persisted and is in a valid GraphML format. However, the underlying `GraphMLTopologyProvider` may perform additional checks or encounters problems while parsing the file. If the Topology does not show up, the `karaf.log` should be checked for any clues what went wrong. In addition it may take a while before the Topology is actually selectable from the Topology UI.

2.2.2. Supported Attributes

A various set of GraphML attributes are supported and interpreted by OpenNMS Meridian while reading the GraphML file. The following table explains the supported attributes and for which GraphML elements they may be used.

The type of the GraphML-Attribute can be either boolean, int, long, float, double, or string. These types are defined like the corresponding types in the Java™-Programming language.

— <http://graphml.graphdrawing.org/primer/graphml-primer.html#Attributes>

Table 1. Supported GraphML Attributes

Property	Required	For element	Type	Default	Description
<code>namespace</code>	yes	Graph	string	-	The namespace must be unique overall existing Topologies.
<code>description</code>	no	Graph	string	-	A description, which is shown in the Info Panel.
<code>preferred-layout</code>	no	Graph	string	D3	Defines a preferred layout.
<code>focus-strategy</code>	no	Graph	string	FIRST	Defines a focus strategy. See Focus Strategies for more information.
<code>focus-ids</code>	no	Graph	string	-	Refers to nodes ids in the graph. This is required if <code>focus-strategy</code> is <code>SPECIFIC</code> . If multiple ids should be add to focus, they are separated by <code>,</code> . Example: <code>node1,node2</code>
<code>semantic-zoom-level</code>	no	Graph	int	1	Defines the default SZL.
<code>vertex-status-provider</code>	no	Graph	string	-	Defines which Vertex Status Provider should be used, e.g. <code>default</code> , <code>script</code> or <code>propagate</code>
<code>iconKey</code>	no	Node	string	generic	Defines the icon. See Icons for more information.

Property	Required	For element	Type	Default	Description
label	no	Graph, Node	string	-	Defines a custom label. If not defined, the <code>id</code> is used instead.
nodeID	no	Node	int	-	Allows referencing the Vertex to an OpenNMS node.
foreignSource	no	Node	string	-	Allows referencing the Vertex to an OpenNMS node identified by foreign source and foreign id. Can only be used in combination with <code>foreignID</code> . Please note that this attribute will not be used when the attribute <code>nodeID</code> is set.
foreignID	no	Node	string	-	Allows referencing the Vertex to an OpenNMS node identified by foreign source and foreign id. Can only be used in combination with <code>foreignSource</code> . Please note that this attribute will not be used when the attribute <code>nodeID</code> is set.
tooltipText	no	Node, Edge	string		Defines a custom tooltip. If not defined, the <code>id</code> attribute is used instead.
level	no	Node	int	0	Sets the level of the Vertex which is used by certain layout algorithms i.e. <code>Hierarchical Layout</code> and <code>Grid Layout</code> .
edge-path-offset	no	Graph, Node	int	20	Controls the spacing between the paths drawn for the edges when there are multiple edges connecting two vertices.
breadcrumb-strategy	no	Graph ML	string	NONE	Defines the breadcrumb strategy to use. See Breadcrumbs for more information.
containerId	no	Graph ML	string	NONE	Defines the id of the container. Is required to access the Graph Service ReST API. If none is provided the value is calculated by joining the graph namespaces using a <code>.</code> as separator.

2.2.3. Focus Strategies

A `Focus Strategy` defines which Vertices should be added to focus when selecting the Topology. The following strategies are available:

- **EMPTY** No Vertex is add to focus.
- **ALL** All Vertices are add to focus.
- **FIRST** The first Vertex is add to focus.
- **SPECIFIC** Only Vertices which id match the graph's property `focus-ids` are added to focus.

2.2.4. Icons

With the *GraphMLTopologyProvider* it is not possible to change the icon from the Topology UI. Instead if a custom icon should be used, each node must contain a `iconKey` property referencing an SVG element.

2.2.5. Vertex Status Provider

The *Vertex Status Provider* calculates the status of the Vertex. There are multiple implementations available which can be configured for each graph: `default`, `script` and `propagate`. If none is specified, there is no status provided at all.

Default Vertex Status Provider

The `default` status provider calculates the status based on the worst unacknowledged alarm associated with the Vertex's node. In order to have a status calculated a (OpenNMS Meridian) node must be associated with the Vertex. This can be achieved by setting the GraphML attribute `nodeID` on the GraphML node accordingly.

Script Vertex Status Provider

The `script` status provider uses scripts similar to the [Edge Status Provider](#). Just place Groovy scripts (with file extension `.groovy`) in the directory `$OPENNMS_HOME/etc/graphml-vertex-status`. All of the scripts will be evaluated and the most severe status will be used for the vertex in the topology's visualization.

If the script shouldn't contribute any status to a vertex just return `null`.

Propagate Vertex Status Provider

The `propagate` status provider follows all links from a node to its connected nodes. It uses the status of these nodes to calculate the status by determining the worst one.

2.2.6. Edge Status Provider

It is also possible to compute a status for each edge in a given graph. Just place Groovy scripts (with file extension `.groovy`) in the directory `$OPENNMS_HOME/etc/graphml-edge-status`. All of the scripts will be evaluated and the most severe status will be used for the edge in the topology's visualization.

The following simple Groovy script example will apply a different style and severity if the edge's associated source node is down.

Scriptable edge status

```
import org.opennms.netmgt.model.OnmsSeverity;
import org.opennms.features.topology.plugins.topo.graphml.GraphMLEdgeStatus;

if ( sourceNode != null && sourceNode.isDown() ) {
    return new GraphMLEdgeStatus(OnmsSeverity.WARNING, [ 'stroke-dasharray' : '5,5',
'stroke' : 'yellow', 'stroke-width' : '6' ]);
} else {
    return new GraphMLEdgeStatus(OnmsSeverity.NORMAL, []);
}
```

If the script shouldn't contribute any status to an edge just return `null`.

2.2.7. Layers

The *GraphMLTopologyProvider* can handle GraphML files with multiple graphs. Each Graph is represented as a Layer in the Topology UI. If a vertex from one graph has an edge pointing to another graph, one can navigate to that layer.

GraphML example defining multiple layers

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
  http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <!-- Key section -->
  <key id="label" for="graphml" attr.name="label" attr.type="string"></key>
  <key id="label" for="graph" attr.name="label" attr.type="string"></key>
  <key id="label" for="node" attr.name="label" attr.type="string"></key>
  <key id="description" for="graph" attr.name="description" attr.type="string"
  ></key>
  <key id="namespace" for="graph" attr.name="namespace" attr.type="string"></key>
  <key id="preferred-layout" for="graph" attr.name="preferred-layout" attr.type=
  "string"></key>
  <key id="focus-strategy" for="graph" attr.name="focus-strategy" attr.type="string"
  "></key>
  <key id="focus-ids" for="graph" attr.name="focus-ids" attr.type="string"></key>
  <key id="semantic-zoom-level" for="graph" attr.name="semantic-zoom-level"
  attr.type="int"/>

  <!-- Label for Topology Selection menu -->
  <data key="label">Layer Example</data>
  <graph id="regions">
    <data key="namespace">acme:regions</data>
    <data key="label">Regions</data>
    <data key="description">The Regions Layer.</data>
    <data key="preferred-layout">Circle Layout</data>
    <data key="focus-strategy">ALL</data>
```

```

<node id="north">
  <data key="label">North</data>
</node>
<node id="west">
  <data key="label">West</data>
</node>
<node id="south">
  <data key="label">South</data>
</node>
<node id="east">
  <data key="label">East</data>
</node>
</graph>
<graph id="markets">
  <data key="namespace">acme:markets</data>
  <data key="description">The Markets Layer.</data>
  <data key="label">Markets</data>
  <data key="description">The Markets Layer</data>
  <data key="semantic-zoom-level">1</data>
  <data key="focus-strategy">SPECIFIC</data>
  <data key="focus-ids">north.2</data>
  <node id="north.1">
    <data key="label">North 1</data>
  </node>
  <node id="north.2">
    <data key="label">North 2</data>
  </node>
  <node id="north.3">
    <data key="label">North 3</data>
  </node>
  <node id="north.4">
    <data key="label">North 4</data>
  </node>
  <node id="west.1">
    <data key="label">West 1</data>
  </node>
  <node id="west.2">
    <data key="label">West 2</data>
  </node>
  <node id="west.3">
    <data key="label">West 3</data>
  </node>
  <node id="west.4">
    <data key="label">West 4</data>
  </node>
  <node id="south.1">
    <data key="label">South 1</data>
  </node>
  <node id="south.2">
    <data key="label">South 2</data>
  </node>

```

```

<node id="south.3">
  <data key="label">South 3</data>
</node>
<node id="south.4">
  <data key="label">South 4</data>
</node>
<node id="east.1">
  <data key="label">East 1</data>
</node>
<node id="east.2">
  <data key="label">East 2</data>
</node>
<node id="east.3">
  <data key="label">East 3</data>
</node>
<node id="east.4">
  <data key="label">East 4</data>
</node>
<!-- Edges in this layer -->
<edge id="north.1_north.2" source="north.1" target="north.2"/>
<edge id="north.2_north.3" source="north.2" target="north.3"/>
<edge id="north.3_north.4" source="north.3" target="north.4"/>
<edge id="east.1_east.2" source="east.1" target="east.2"/>
<edge id="east.2_east.3" source="east.2" target="east.3"/>
<edge id="east.3_east.4" source="east.3" target="east.4"/>
<edge id="south.1_south.2" source="south.1" target="south.2"/>
<edge id="south.2_south.3" source="south.2" target="south.3"/>
<edge id="south.3_south.4" source="south.3" target="south.4"/>
<edge id="north.1_north.2" source="north.1" target="north.2"/>
<edge id="north.2_north.3" source="north.2" target="north.3"/>
<edge id="north.3_north.4" source="north.3" target="north.4"/>

<!-- Edges to different layers -->
<edge id="west_north.1" source="north" target="north.1"/>
<edge id="north_north.2" source="north" target="north.2"/>
<edge id="north_north.3" source="north" target="north.3"/>
<edge id="north_north.4" source="north" target="north.4"/>
<edge id="south_south.1" source="south" target="south.1"/>
<edge id="south_south.2" source="south" target="south.2"/>
<edge id="south_south.3" source="south" target="south.3"/>
<edge id="south_south.4" source="south" target="south.4"/>
<edge id="east_east.1" source="east" target="east.1"/>
<edge id="east_east.2" source="east" target="east.2"/>
<edge id="east_east.3" source="east" target="east.3"/>
<edge id="east_east.4" source="east" target="east.4"/>
<edge id="west_west.1" source="west" target="west.1"/>
<edge id="west_west.2" source="west" target="west.2"/>
<edge id="west_west.3" source="west" target="west.3"/>
<edge id="west_west.4" source="west" target="west.4"/>
</graph>
</graphml>

```

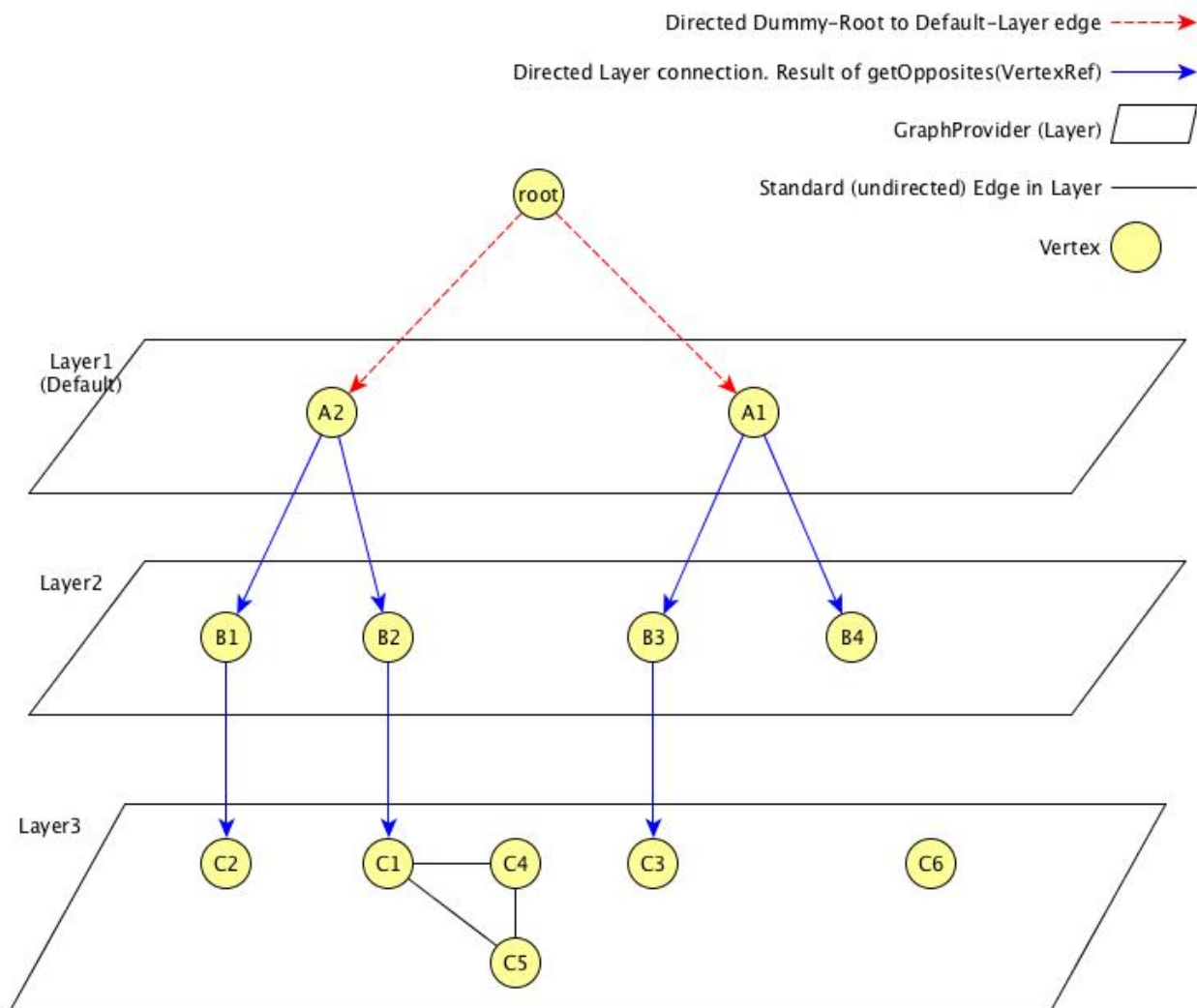
2.2.8. Breadcrumbs

When multiple Layers are used it is possible to navigate between them (**navigate to** option from vertex' context menu). To give the user some orientation breadcrumbs can be enabled with the **breadcrumb-strategy** property.

The following strategies are supported:

- **NONE** No breadcrumbs are shown.
- **SHORTEST_PATH_TO_ROOT** generates breadcrumbs from all visible vertices to the root layer (TopologyProvider). The algorithm assumes a hierarchical graph. Be aware, that all vertices **MUST** share the same root layer, otherwise the algorithm to determine the path to root does not work.

The following figure visualizes a graphml defining multiple layers (see below for the graphml definition).



From the given example, the user can select the **Breadcrumb Example** Topology Provider from the menu. The user can switch between **Layer 1**, **Layer 2** and **Layer 3**. In addition for each vertex which has connections to another layer, the user can select the **navigate to** option from the context menu of that vertex to navigate to the according layer. The user can also search for all vertices and add it

to focus.

The following behaviour is implemented:

- If a user navigates from one vertex to a vertex in another layer, the view is switched to that layer and adds all vertices to focus, the `source vertex` pointed to. The Breadcrumb is `<parent layer name> > <source vertex>`. For example, if a user navigates from `Layer1:A2` to `Layer2:B1` the view switches to `Layer 2` and adds `B1` and `B2` to focus. In addition `Layer 1 > A2` is shown as Breadcrumbs.
- If a user directly switches to another layer, the default focus strategy is applied, which may result in multiple vertices with no unique parent. The calculated breadcrumb is: `<parent layer name> > Multiple <target layer name>`. For example, if a user switches to `Layer 3`, all vertices of that layer are added to focus (`focus-strategy=ALL`). No unique path to root is found, the following breadcrumb is shown instead: `Layer 1 > Multiple Layer 1 > Multiple Layer 2`
- If a user adds a vertex to focus, which is not in the current selected layer, the view switches to that layer and only the "new" vertex is added to focus. The generated breadcrumb shows the path to root through all layers. For example, the user adds `C3` to focus, and the current layer is `Layer 1`, then the generated breadcrumb is as follows: `Layer 1 > A1 > B3`.
- Only elements between layers are shown in the breadcrumb. Connections on the same layer are ignored. For example, a user adds `C5` to focus, the generated breadcrumb is as follows: `Layer 1 > A2 > B2`

The following graphml file defines the above shown graph. Be aware, that the root vertex shown above is generated to help calculating the path to root. It must not be defined in the graphml document.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
    http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <key id="breadcrumb-strategy" for="graphml" attr.name="breadcrumb-strategy"
  attr.type="string"></key>
  <key id="label" for="all" attr.name="label" attr.type="string"></key>
  <key id="description" for="graph" attr.name="description" attr.type="string"
  ></key>
  <key id="namespace" for="graph" attr.name="namespace" attr.type="string"></key>
  <key id="focus-strategy" for="graph" attr.name="focus-strategy" attr.type="string"
  "></key>
  <key id="focus-ids" for="graph" attr.name="focus-ids" attr.type="string"></key>
  <key id="preferred-layout" for="graph" attr.name="preferred-layout" attr.type="
  string"></key>
  <key id="semantic-zoom-level" for="graph" attr.name="semantic-zoom-level"
  attr.type="int"/>
  <data key="label">Breadcrumb Example</data>
  <data key="breadcrumb-strategy">SHORTEST_PATH_TO_ROOT</data>
  <graph id="L1">
    <data key="label">Layer 1</data>
    <data key="namespace">acme:layer1</data>
```

```

<data key="focus-strategy">ALL</data>
<data key="preferred-layout">Circle Layout</data>
<node id="a1">
  <data key="label">A1</data>
</node>
<node id="a2">
  <data key="label">A2</data>
</node>
<edge id="a1_b3" source="a1" target="b3"/>
<edge id="a1_b4" source="a1" target="b4"/>
<edge id="a2_b1" source="a2" target="b1"/>
<edge id="a2_b2" source="a2" target="b2"/>
</graph>
<graph id="L2">
  <data key="label">Layer 2</data>
  <data key="focus-strategy">ALL</data>
  <data key="namespace">acme:layer2</data>
  <data key="preferred-layout">Circle Layout</data>
  <data key="semantic-zoom-level">0</data>
  <node id="b1">
    <data key="label">B1</data>
  </node>
  <node id="b2">
    <data key="label">B2</data>
  </node>
  <node id="b3">
    <data key="label">B3</data>
  </node>
  <node id="b4">
    <data key="label">B4</data>
  </node>
  <edge id="b1_c2" source="b1" target="c2"/>
  <edge id="b2_c1" source="b2" target="c1"/>
  <edge id="b3_c3" source="b3" target="c3"/>
</graph>
<graph id="Layer 3">
  <data key="label">Layer 3</data>
  <data key="focus-strategy">ALL</data>
  <data key="description">Layer 3</data>
  <data key="namespace">acme:layer3</data>
  <data key="preferred-layout">Grid Layout</data>
  <data key="semantic-zoom-level">1</data>
  <node id="c1">
    <data key="label">C1</data>
  </node>
  <node id="c2">
    <data key="label">C2</data>
  </node>
  <node id="c3">
    <data key="label">C3</data>
  </node>

```



```

<node id="c4">
  <data key="label">C4</data>
</node>
<node id="c5">
  <data key="label">C5</data>
</node>
<node id="c6">
  <data key="label">C6</data>
</node>
<edge id="c1_c4" source="c1" target="c4"/>
<edge id="c1_c5" source="c1" target="c5"/>
<edge id="c4_c5" source="c4" target="c5"/>
</graph>
</graphml>

```

2.3. Topologies Updates

In OpenNMS Meridian the `Kafka Producer` uses Topologies Updates to send topologies messages to Kafka.

Enhanced Linkd has six `OnmsTopologyUpdater`s each sending its own NODES, BRIDGE, CDP, ISIS, LLDP and OSPF protocol `OnmsTopologyMessage`.

`Kafka Producer` implements an `OnmsTopologyConsumer` that reads the `OnmsTopologyMessage` to be shared over Kafka.

The Topologies updates api are available under `opennms/features/topologies`

2.3.1. OnmsTopologyUpdater

To set up an implementation class of `OnmsTopologyUpdater` interface it must be implemented three methods:

```
OnmsTopology getTopology();
```

```
OnmsTopologyProtocol getProtocol() throws OnmsTopologyException;
```

```
String getName();
```

An `OnmsTopologyUpdater` holds the `OnmsTopology` of the specified `OnmsTopologyProtocol`.

In order to send `OnmsTopologyMessage` to subscribers the updater should register and send messages using the provided methods into `OnmsTopologyDao`

To register use:

```
register(OnmsTopologyUpdater updater) Only one updater for protocol is allowed to register over OnmsTopologyDao.
```

To send messages to subscriber use:

```
void update(OnmsTopologyUpdater updater, OnmsTopologyMessage message) throws
```

```
OnmsTopologyException;
```

To unregister use:

```
unregister(OnmsTopologyUpdater updater)
```

Registering Updaters with OnmsTopologyDao will also provide Topology using `OnmsTopologyDao`

to get a Topology for a specific protocol use:

```
OnmsTopology getTopology(String protocol)
```

2.3.2. OnmsTopologyRef

`OnmsTopologyRef` is the topology basic interface. All topology defined objects are implementing it.

The methods defining OnmsTopologyRef are:

The Id: a unique String identifier of the topology Object.

```
String getId();
```

The tool tip text: some additional information to describe the topology object.

```
String getToolTipText();
```

2.3.3. OnmsTopologyMessage

`OnmsTopologyMessage` is defined by `OnmsTopologyProtocol` by a `TopologyMessageStatus` and by an `OnmsTopologyRef` and it is send to all subscriber for the specified protocol by the registered updater via the `OnmsTopologyDao`

2.3.4. OnmsTopologyProtocol

`OnmsTopologyProtocol` represent a protocol. It is defined by a String but it is not case sensitive, CDP either cdp and Cdp will define the same protocol.

An `OnmsTopologyUpdater` can be an updater for only one protocol.

An `OnmsTopologyConsumer` can be a consumer for several protocols.

2.3.5. TopologyMessageStatus

`TopologyMessageStatus` is an enum whose allowed values are: UPDATE or DELETE.

UPDATE message means that the topology message holds information about an `OnmsTopologyRef` still available in Topology

DELETE message means that the object `OnmsTopologyRef` has been deleted from topology.

2.3.6. OnmsTopology

A topology is made by vertices and edges each of uniquely identified by an `OnmsTopologyRef`

`OnmsTopology` is a SET of `OnmsTopologyVertex` and a SET of `OnmsTopologyEdge`.

Also is possible to set and get a Default Vertex.

2.3.7. OnmsTopologyVertex

`OnmsTopologyVertex` implements `OnmsTopologyRef` and represents a vertex

2.3.8. OnmsTopologyEdge

`OnmsTopologyEdge` implements `OnmsTopologyRef` and represents an Edge and is made by a two `OnmsTopologyPort` the source and the target of the edge.

2.3.9. OnmsTopologyPort

`OnmsTopologyPort` implements `OnmsTopologyRef` represents a Port belonging to a Vertex so has a property to set and get the `OnmsTopologyVertex`.

The Port is an element of the Topology because it belongs to an Edge and is the connection element.

Two vertices are connected by an Edge by the corresponding Port.

2.3.10. OnmsTopologyConsumer

To set up an implementation class of `OnmsTopologyConsumer` interface it must be implemented three methods:

```
String getName();  
Set<OnmsTopologyProtocol> getProtocols();  
void consume(OnmsTopologyMessage message);
```

An `OnmsTopologyconsumer` consume the `OnmsTopologyMessage` of the specified list `OnmsTopologyProtocol` provided by the `getProtocols()` method implementing the method `void consume(OnmsTopologyMessage message)`

In order to receive `OnmsTopologyMessage` from the registered "Updaters", the "Consumer" should subscribe `OnmsTopologyDao`

To subscribe use:

```
void subscribe(OnmsTopologyConsumer consumer)
```

To unsubscribe use:

```
void unsubscribe(OnmsTopologyConsumer consumer)
```

Chapter 3. Graph Service API

This section covers the insides of the new *Graph Service API*. The *Graph Service API* was heavily inspired by the [GraphML](#) model and the original Topology API.

3.1. Terminology

A **Graph** consists of any number of points with any number of connections in between. Usually these points are called **Nodes**, but as **Nodes** have a specific meaning in *OpenNMS Meridian* in the context of the *Graph Service* they are called **Vertices** (plural of **Vertex**). The connection between two **Vertices** are usually called **Link**, however it also has a specific meaning inside *OpenNMS Meridian*, thus it is called **Edge**. A **Graph Container** contains any number of **Graphs**. In general an **Element** can either be a **Graph**, **Vertex** or **Edge**.

3.2. Graph Model

3.2.1. Generic Graph Model

The *Generic Graph Model* is very similar to [GraphML](#)'s model and consists of the basic elements:

- `GenericGraphContainer`,
- `GenericGraph`,
- `GenericVertex` and
- `GenericEdge`

Each of the above have besides their designated functionality described in [Terminology](#) also properties assigned to them. Some properties are mandatory and determined by the element's type. Others are optionally and totally up to the entity providing a graph to the system.

The following rules apply to the *Generic Graph Model*:

- Each Graph container must be uniquely identified. This identifier is called a `container id`. The `container id` must be unique overall containers and graph namespaces.
- Each Graph must be uniquely identified over all graphs (meaning the namespace must be unique throughout the system). This identifier is called a `namespace`. Each `namespace` must be unique overall container ids and graph namespaces.
- Each Vertex and Edge must be uniquely identified by an identifier. This identifier is called an `id`. The `id` must not be unique in the system, but within the graph.
- Each Vertex and Edge have the same `namespace` as the Graph they are part of. This allows to uniquely identify vertices and edges throughout the system.
- Each Graph, Vertex or Edge have additional properties to define their nature in more detail. E.g. a Vertex may contain a `label`, `node` or `location` property.
- Each Edge contains the source and target id of the Vertex it connects, therefore is ALWAYS directed.

- Each edge may reference Vertices from a different Graph (`namespace`). However at least one side must share the same `namespace` as the edge itself. This is the "owning" side, thus the edge will be a member of that Graph.



A graph may define a property `id` besides the property `namespace`. This is mostly only relevant, when importing graphs via GraphML as each graph element must have an `id` attribute set. When importing such GraphML each imported graph will have a `namespace` and `id` property set. In general, it is recommended, that the value of the attribute `id` SHOULD be identically to the `namespace` property even if it MUST NOT.

Refer to module `features/graph/api` in package `org.opennms.netmgt.graph.api.generic` for implementation details.

3.2.2. Domain Graph Model

A *Domain Graph Model* is a model implementing a use case in its domain using the *Generic Graph Model*. This can be achieved by simply using the *Generic Graph Model* and setting the according properties. For more complex use cases a more Java-friendly way - Basic Domain objects are available - to act as a "view" on the *Generic Graph Model* to provide a more convenient implementation. These domain objects don't hold information on their own but provide only access to information that can be deduced from the underlying Generic Model and it's properties.



For the *Domain Graph Model* the same rules apply as for the [Generic Graph Model](#).

See `features/graph/provider/bsm` for a domain model using the basic domain graph classes (`AbstractDomainGraph`, `AbstractDomainVertex`, etc).

See `features/graph/provider/graphml` for a domain model using the Generic Graph Model classes.

3.2.3. Immutability

The graph container and all it's elements are immutable once created. In order to fully create a container various builders are provided.

3.3. Providers

To get a new graph or graph container to the system either a `GraphContainerProvider` or `GraphProvider` must be implemented and exposed via the OSGi-registry. The latter is a convenient way of exposing a container which only consists of one graph.

3.3.1. Meta Information

Loading a graph container or graph may be very expensive. Especially, if only the meta data are of interest (e.g. label, description). Therefore each provider allows to fetch the meta data only without loading the graph or container.

3.4. Cache

All calls to `GraphContainerProvider.loadGraphContainer()` are cached until *OpenNMS Meridian* restarts. This means it is up to the implementor to invalidate the cache.

This can be achieved by manually invoking the `org.opennms.netmgt.graph.api.service.GraphContainerCache.invalidate(String containerId)` method as the `GraphContainerCache` is exposed as an OSGi-service. Alternatively a service property can be used to invalidate a cache entry after it was first populated to the cache:

```
<service interface="org.opennms.netmgt.graph.api.service.GraphProvider" ref=
"applicationGraphProvider">
  <service-properties>
    <entry key="cacheInvalidateInterval" value="300" /> <!-- seconds -->
  </service-properties>
</service>
```

3.5. Search

The goal behind searching is usually to bring one or more vertices into the focus of the user. Usually the user searches for something like a node label or category name. Mostly, those things do not directly relate to any element in the graph.

Therefore a search always consists of the following process:

1. Search for a concrete item in the system (e.g. node, vertex, category)
2. Resolve all vertices which match the concrete item.

Internally `SearchProvider` are used which allow to first make suggestions based on an input. A `NodeSearchProvider` might return a list of suggestions representing nodes in the system. The `NodeSearchProvider` knows how to "resolve" that selection to a unique set of vertices after the user selected one.



At the moment searching is ALWAYS performed in the context of a given namespace. Searching over multiple namespaces is not supported or encouraged.

3.6. Focus / Semantic Zoom Level / View

The focus itself is a list of vertices within the graph. The *Semantic Zoom Level* (or *szl*) indicates the number of hops (= distance) a vertex may be away from any vertex within the focus to be shown. If the *Semantic Zoom Level* is `0` only the vertices in focus are shown.

The applied *Semantic Zoom Level* and *Focus* is called a *View* of the graph.

If the focus contains elements which are not part of the graph, they are not shown.

The *Graph Service API* allows to create a view on any given graph considering a custom or default

focus as well as the semantic zoom level. If no focus is provided when requesting the view, the default focus is applied.

3.7. Listening for Changes

It is possible to listen for graph or graph container changes. Various listeners can be exposed as an OSGi-service.

The calculation of changes must be triggered manually. To do so, use the `org.opennms.netmgt.graph.api.updates.GraphNotificationService` service, which is exposed as an OSGi service

Refer to package `org.opennms.netmgt.graph.api.updates` in module `features/graph/api` for available options.

3.8. Persistence

Each graph or graph container can be persisted to the database using the `org.opennms.netmgt.graph.api.persistence.GraphRepository`.

3.9. Enrichment

Enrichment is the process to enrich the view of the graph with additional information, usually used when the enriched data is expensive to load/calculate, e.g. load node data or calculate the status.

Due to performance considerations, enrichment only works on the view of a graph and cannot be performed on the full graph.

3.9.1. Build in Enrichment

Node Enrichment

By default all vertices can be enriched with node information if *Node Enrichment* is enabled. To enable the enrichment of node information, the property `enrichment.resolveNodes` must be set to `true` on the graph.

Afterwards each vertex which either has a `nodeID` (`integer`) or `foreignSource` (`string`) and `foreignID` (`string`) property assigned will be enriched if a node with that information is found in the system. A shorter version in form of `nodeCriteria=<foreignSource>:<foreignID>` is also available.

3.10. ReST API

The *Graph Service API* provides a ReST API which is documented in detail [here](#).

The Graph Service API ReST endpoints serializes the requested graph container, graph or view in json. As the container and each element within that container may contain custom properties, the type of the property is not static. This means, each `GraphContainerProvider` (or `GraphProvider`) can set properties on any element using a type which is only known by the domain the graph container is

created in. However when serializing this object as JSON it is not clear to the ReST endpoint how to do that. In order to allow these values to be set as properties anyways, a custom `org.opennms.netmgt.graph.rest.api.PropertyConverter` can be exposed as an OSGi service.

3.11. Limitations

The following limitations are known for the *Graph Service API*:

- Status of vertices is not exposed
- custom images/icons cannot be set
- custom edge/vertex status providers are not implemented
- VMWare Topology Provider not fully migrated
- EnhancedLinkd Topology Provider not fully migrated

Chapter 4. CORS Support

4.1. 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](#)

4.2. 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 Meridian

4.3. 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 5. 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 Meridian.

5.1. 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.

5.2. 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.

5.3. 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. XML is delivered without namespaces. Please note: If a namespace is added manually in order to use a XML tool to validate against the XSD (like xmllint) it won't be preserved when OpenNMS updates that file. The same applies to comments. To get JSON encoded responses one has to send the following header with the request: `Accept: application/json`.

5.4. Standard Parameters

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

Table 2. 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

Param	Description
eter	
<p>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 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:</p>	
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). **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)

5.5. 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=nonnull	would return the first 10 events that have a non-null Ack time (i.e. those that have been acknowledged)
/events?eventAckTime=nonnull&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 nonnull value causes the comparator to be ignored for eventAckTime

Resource	Description
<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)

5.6. HTTP Return Codes

The following apply for OpenNMS Horizon 18 and newer.

- DELETE requests are going to return a 202 (ACCEPTED) if they are performed asynchronously otherwise they return a 204 (NO_CONTENT) on success.
- All the PUT requests are going to return a 204 (NO_CONTENT) on success.
- All the POST requests that can either add or update an entity are going to return a 204 (NO_CONTENT) on success.
- All the POST associated to resource addition are going to return a 201 (CREATED) on success.
- All the POST requests where it is required to return an object will return a 200 (OK).
- All the requests excepts GET for the Requisitions end-point and the Foreign Sources Definitions end-point will return 202 (ACCEPTED). This is because all the requests are actually executed asynchronously and there is no way to know the status of the execution, or wait until the processing is done.
- If a resource is not modified during a PUT request, a NOT_MODIFIED will be returned. A NO_CONTENT will be returned only on a success operation.
- All GET requests are going to return 200 (OK) on success.
- All GET requests are going to return 404 (NOT_FOUND) when a single resource doesn't exist; but will return 400 (BAD_REQUEST), if an intermediate resource doesn't exist. For example, if a specific IP doesn't exist on a valid node, return 404. But, if the IP is valid and the node is not valid, because the node is an intermediate resource, a 400 will be returned.
- If something not expected is received from the Service/DAO Layer when processing any HTTP request, like an exception, a 500 (INTERNAL_SERVER_ERROR) will be returned.
- Any problem related with the incoming parameters, like validations, will generate a 400 (BAD_REQUEST).

5.7. Identifying Resources

Some endpoints deal in resources, which are identified by *Resource IDs*. Since every resource is ultimately parented under some node, identifying the node which contains a resource is the first step in constructing a resource ID. Two styles are available for identifying the node in a resource ID:

Style	Description	Example
<code>node[ID]</code>	Identifies a node by its database ID, which is always an integer	<code>node[42]</code>
<code>node[FS:FID]</code>	Identifies a node by its foreign-source name and foreign-ID, joined by a single colon	<code>node[Servers:115da833-0957-4471-b496-a731928c27dd]</code>

The node identifier is followed by a period, then a resource-type name and instance name. The instance name's characteristics may vary from one resource-type to the next. A few examples:

Value	Description
<code>nodeSnmpp[]</code>	Node-level (scalar) performance data for the node in question. This type is the only one where the instance identifier is empty.
<code>interfaceSnmpp[eth0-04013f75f101]</code>	A layer-two interface as represented by a row in the SNMP <code>ifTable</code> . The instance identifier is composed of the interface's <code>ifName</code> and its <code>ifPhysAddress</code> (if it has one).
<code>dskIndex[_root_fs]</code>	The root filesystem of a node running the Net-SNMP management agent.

Putting it all together, here are a few well-formed resource IDs:

- `node[1].nodeSnmpp[]`
- `node[42].interfaceSnmpp[eth0-04013f75f101]`
- `node[Servers:115da833-0957-4471-b496-a731928c27dd].dskIndex[_root_fs]`

5.8. Expose ReST services via OSGi

In order to expose a ReST service via OSGi the following steps must be followed:

1. Define an interface, containing java jax-rs annotations
2. Define a class, implementing that interface
3. Create an OSGi bundle which exports a service with the interface from above

5.8.1. Define a ReST interface

At first a public interface must be created which must contain jax-rs annotations.

```

@Path("/datachoices") ①
public interface DataChoiceRestService {

    @POST ②
    void updateCollectUsageStatisticFlag(@Context HttpServletRequest request,
    @QueryParam("action") String action);

    @GET
    @Produces(value={MediaType.APPLICATION_JSON})
    UsageStatisticsReportDTO getUsageStatistics();
}

```

① Each ReST interface must either have a `@Path` or `@Provider` annotation. Otherwise it is not considered a ReST service.

② Use jax-rs annotations, such as `@Post`, `@GET`, `@PUT`, `@Path`, etc. to define the ReST service.

5.8.2. Implement a ReST interface

A class must implement the ReST interface.



The class may or may not repeat the jax-rs annotations from the interface. This is purely for readability. Changing or adding different jax-rs annotations on the class, won't have any effect.

```

public class DataChoiceRestServiceImpl implements DataChoiceRestService {

    @Override
    public void updateCollectUsageStatisticFlag(HttpServletRequest request, String
action) {
        // do something
    }

    @Override
    public UsageStatisticsReportDTO getUsageStatistics() {
        return null;
    }
}

```

5.8.3. Export the ReST service

At last the ReST service must be exported via the bundlecontext. This can be either achieved using an Activator or the blueprint mechanism.

```

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.osgi.org/xmlns/blueprint/v1.0.0
    http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd"
">
  <bean id="dataChoiceRestService" class=
"org.opennms.features.datachoices.web.internal.DataChoiceRestServiceImpl" /> ①
  <service interface="org.opennms.features.datachoices.web.DataChoiceRestService"
ref="dataChoiceRestService" > ②
    <service-properties>
      <entry key="application-path" value="/rest" /> ③
    </service-properties>
  </service>
</blueprint>

```

- ① Create the ReST implementation class
- ② Export the ReST service
- ③ Define where the ReST service will be exported to, e.g. `/rest`, `/api/v2`, but also completely different paths can be used. If not defined, `/services` is used.

For a full working example refer to the [datachoices feature](#).

5.9. Currently Implemented Interfaces

5.9.1. Acknowledgements



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 <i>XML</i> or <i>JSON</i> .)
<code>/acks/{id}</code>	Get the acknowledgement specified by the given <i>ID</i> .

POSTs (Setting Data)

Resource	Description
/acks	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
```

5.9.2. 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 <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>).

5.9.3. Alarms



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.

Resource	Description
<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, <i>NORMAL</i> → <i>MINOR</i> , <i>MAJOR</i> → <i>CRITICAL</i> , 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.

v2 API

Since Horizon 20.1.0, an alarms endpoint has been available in the v2 API that uses [FIQL](#) for querying.

Examples:

- `/api/v2/alarms?_s=alarm.reductionKey==uei.opennms.org/nodes/nodeDown::*`

- `/api/v2/alarms?_s=alarm.uei==uei.opennms.org/nodes/nodeDown`
- `/api/v2/alarms?_s=alarm.ipAddr==127.0.0.1`
- `/api/v2/alarms?_s=node.label==localhost`



If your query string includes a FIQL reserved character like a comma (,) or a semicolon (;) you need to double percent-escape it.

Suppose you have a service with a comma in the name, like `COMMA,SERVICE` and you want to retrieve all the alarms for its failure. Your query would look like `/api/v2/alarms?_s=alarm.reductionKey==uei.opennms.org/nodes/nodeLostService::*:COMMA%252CSERVICE`.

5.9.4. Alarm History

The *Alarm History API* can be used to retrieve state changes and point-in-time snapshots for alarms.



This endpoint is only available when the *Alarm History* feature is enabled. See the Administrators Guide for details.

GETs (Reading Data)

Resource	Description
<code>/alarms/history/{alarmId}/states</code>	Retrieve the complete set of state changes for the given alarm.
<code>/alarms/history/{alarmId}</code>	Retrieve the last known state of an alarm at the given time.
<code>/alarms/history</code>	Retrieve the last known state of all alarms which were active at the given time.

name	default	comment
matchType	<code>alarm-id</code>	By default we lookup alarms based on the given database id. This property can be used to change the matching behaviour to lookup by reduction key instead.
at	(current time)	Timestamp in milliseconds.

5.9.5. 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 <i>XML</i> or <i>JSON</i> .)
<code>/events/{id}</code>	Get the event specified by the given <i>ID</i> .

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.



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

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

5.9.6. Categories

GETs (Reading Data)

Resource	Description
<code>/categories</code>	Get all configured categories.
<code>/categories/{category}</code>	Get the category specified by the given name.
<code>/categories/{category}/nodes/{node}</code>	Get the category specified by the given name for the given node (similar to <code>/nodes/{node}/categories/{category}</code>)
<code>/categories/nodes/{node}</code>	Get the categories for a given node (similar to <code>/nodes/{node}/categories</code>)
<code>/categories/groups/{group}</code>	Get the categories for a given user group (similar to <code>/groups/{group}/categories</code>)

POSTs (Adding Data)

Resource	Description
<code>/categories</code>	Adds a new category.

PUTs (Modifying Data)

Resource	Description
<code>/categories/{category}</code>	Update the specified category
<code>/categories/{category}/nodes/{node}</code>	Modify the category with the given node ID and name (similar to <code>/nodes/{node}/categories/{category}</code>)
<code>/categories/{category}/groups/{group}</code>	Add the given category to the given user group (similar to <code>/groups/{group}/categories/{category}</code>)

DELETES (Removing Data)

Resource	Description
<code>/categories/{category}</code>	Delete the specified category
<code>/categories/{category}/nodes/{node}</code>	Remove the given category from the given node (similar to <code>/nodes/{node}/categories/{category}</code>)
<code>/categories/{category}/groups/{group}</code>	Remove the given category from the given user group (similar to <code>/groups/{group}/categories/{category}</code>)

5.9.7. Flow API

The *Flow API* can be used to retrieve summary statistics and time series data derived from persisted flows.



Unless specific otherwise, all unit of time are expressed in milliseconds.

GETs (Reading Data)

Resource	Description
<code>/flows/count</code>	Retrieve the number of flows available
<code>/flows/exporters</code>	Retrieve basic information for the exporter nodes that have flows available
<code>/flows/exporters/{nodeCriteria}</code>	Retrieve detailed information about a specific exporter node
<code>/flows/applications</code>	Retrieve traffic summary statistics for the top N applications or specific applications

Resource	Description
/flows/applications/enumerate	Retrieve a list of the applications with flows
/flows/applications/series	Retrieve time series metrics for the top N applications or specific applications
/flows/conversations	Retrieve traffic summary statistics for the top N conversations or specific conversations
/flows/conversations/enumerate	Retrieve a list of the conversations with flows
/flows/conversations/series	Retrieve time series metrics for the top N conversations or specific conversations
/flows/hosts	Retrieve traffic summary statistics for the top N hosts or specific hosts
/flows/hosts/enumerate	Retrieve a list of the hosts with flows
/flows/hosts/series	Retrieve time series metrics for the top N hosts or specific hosts

All of the endpoints support the following query string parameters to help filter the results:



The given filters are combined using a logical **AND**. There is no support for using **OR** logic, or combinations thereof.

name	default	comment
start	-14400000	Timestamp in milliseconds. If > 0, the timestamp is relative to the UNIX epoch (January 1st 1970 00:00:00 AM). If < 0, the timestamp is relative to the end option (i.e.: default value is 4 hours ago).
end	0	Timestamp in milliseconds. If <= 0, the effective value will be the current timestamp.
ifIndex	(none)	Filter for flows that came in through the given SNMP interface.

exporterNode	(none)	Filter for flows that came where exported by the given node. Support either node id (integer) i.e. 1, or foreign source and foreign id lookups i.e. FS:FID.
--------------	--------	--

The `exporters` endpoints do not support any parameters.

The `applications/enumerate` endpoint also supports:

name	default	comment
limit	10	Number of applications with flows to return
prefix	""	If provided, filters the results to only include applications with names that start with the given prefix (using fuzzy matching)

The `applications` and `applications/series` endpoints also support:

name	default	comment
N	none	Number of top entries (determined by total bytes transferred) to return
includeOther	false	When set to <code>true</code> the results will also include an additional row or column that contains the bytes transferred for the flows that fall outside of the selected set
application	none	If provided, returns only flow series that match the given application(s). This field can be repeated for any number of applications to be included in the query.

Both endpoints require one of `N` or `application` query parameters to be set and will return an error if neither or both are set.

The `conversations/enumerate` endpoint also supports:

name	default	comment
limit	10	Number of hosts with flows to return
location	".*"	If provided, filters the results to only include conversations matching the given location regex pattern
protocol	".*"	If provided, filters the results to only include conversations matching the given protocol regex pattern
lower	".*"	If provided, filters the results to only include conversations matching the given lower IP address regex pattern
upper	".*"	If provided, filters the results to only include conversations matching the given upper IP address regex pattern
application	".*"	If provided, filters the results to only include conversations matching the given application regex pattern

The `conversations` and `conversations/series` endpoints also supports:

name	default	comment
N	10	Number of top entries (determined by total bytes transferred) to return
includeOther	false	When set to <code>true</code> the results will also include an additional row or column that contains the bytes transferred for the flows that fall outside of selected set.
conversation	none	If provided, returns only flow series that match the given conversation(s). This field can be repeated for any number of conversations to be included in the query.

name	default	comment
hostname_mode	"replace"	Controls how IP addresses are replaced with hostnames if available: <ul style="list-style-type: none"> • replace will replace the IP with the hostname • append will append the hostname to the IP address • hide will not show the hostname at all



Make sure to URL encode the conversation value in your request.

Both endpoints requires one of **N** or **conversation** query parameters to be set and will return an error if neither or both are set.

The **conversations/series** endpoint also supports:

name	default	comment
step	300000	Requested time interval between rows.

The **hosts/enumerate** endpoint also supports:

name	default	comment
limit	10	Number of hosts with flows to return
pattern	".*"	If provided, filters the results to only include hosts with names that match the given regex pattern

The **hosts** and **hosts/series** endpoints also support:

name	default	comment
N	none	Number of top entries (determined by total bytes transferred) to return

name	default	comment
includeOther	false	When set to <code>true</code> the results will also include an additional row or column that contains the bytes transferred for the flows that fall outside of the selected set.
host	none	If provided, returns only flow series that match the given host(s). This field can be repeated for any number of hosts to be included in the query.

Both endpoints requires one of `N` or `host` query parameters to be set and will return an error if neither or both are set.

Examples

Retrieve the number of flows persisted in the last 4 hours

```
curl -u admin:admin http://localhost:8980/opennms/rest/flows/count
```

Response

```
915
```

Retrieve traffic summary for top 10 applications in the last 4 hours

```
curl -u admin:admin http://localhost:8980/opennms/rest/flows/applications
```

Response

```
{
  "start": 1513788044417,
  "end": 1513802444417,
  "headers": ["Application", "Bytes In", "Bytes Out"],
  "rows": [
    ["https", 48789, 136626],
    ["http", 12430, 5265]
  ]
}
```

Retrieve traffic summary for top 10 conversations in the last 4 hours

```
curl -u admin:admin http://localhost:8980/opennms/rest/flows/conversations
```

Response

```
{
  "start": 1513788228224,
  "end": 1513802628224,
  "headers": ["Location", "Protocol", "Source IP", "Source Port", "Dest. IP", "Dest.
Port", "Bytes In", "Bytes Out"],
  "rows": [
    ["Default", 17, "10.0.2.15", 33816, "172.217.0.66", 443, 12166, 117297],
    ["Default", 17, "10.0.2.15", 32966, "172.217.0.70", 443, 5042, 107542],
    ["Default", 17, "10.0.2.15", 54087, "172.217.0.67", 443, 55393, 5781],
    ["Default", 17, "10.0.2.15", 58046, "172.217.0.70", 443, 4284, 46986],
    ["Default", 6, "10.0.2.15", 39300, "69.172.216.58", 80, 969, 48178],
    ["Default", 17, "10.0.2.15", 48691, "64.233.176.154", 443, 8187, 39847],
    ["Default", 17, "10.0.2.15", 39933, "172.217.0.65", 443, 1158, 33913],
    ["Default", 17, "10.0.2.15", 60751, "216.58.218.4", 443, 5504, 24957],
    ["Default", 17, "10.0.2.15", 51972, "172.217.0.65", 443, 2666, 22556],
    ["Default", 6, "10.0.2.15", 46644, "31.13.65.7", 443, 459, 16952]
  ]
}
```

Retrieve time series data for top 3 applications in the last 4 hours

```
curl -u admin:admin
http://localhost:8980/opennms/rest/flows/applications/series?N=3&includeOther=true&ste
p=3600000
```

Response

```
{
  "start": 1516292071742,
  "end": 1516306471742,
  "columns": [
    {
      "label": "domain",
      "ingress": true
    },
    {
      "label": "https",
      "ingress": true
    },
    {
      "label": "http",
      "ingress": true
    },
    {
      "label": "Other",
      "ingress": true
    }
  ],
  "timestamps": [
    1516291200000,
    1516294800000,
    1516298400000
  ],
  "values": [
    [9725, 12962, 9725],
    [70665, 125044, 70585],
    [10937, 13141, 10929],
    [1976, 2508, 2615]
  ]
}
```

Retrieve time series data for top 3 conversations in the last 4 hours

```
curl -u admin:admin
http://localhost:8980/opennms/rest/flows/conversations/series?N=3&step=3600000
```

Response

```
{
  "start": 1516292150407,
  "end": 1516306550407,
  "columns": [
    {
      "label": "10.0.2.15:55056 <-> 152.19.134.142:443",
      "ingress": false
    },
    {
      "label": "10.0.2.15:55056 <-> 152.19.134.142:443",
      "ingress": true
    },
    {
      "label": "10.0.2.15:55058 <-> 152.19.134.142:443",
      "ingress": false
    },
    {
      "label": "10.0.2.15:55058 <-> 152.19.134.142:443",
      "ingress": true
    },
    {
      "label": "10.0.2.2:61470 <-> 10.0.2.15:8980",
      "ingress": false
    },
    {
      "label": "10.0.2.2:61470 <-> 10.0.2.15:8980",
      "ingress": true
    }
  ],
  "timestamps": [
    1516294800000,
    1516298400000
  ],
  "values": [
    [17116, "NaN"],
    [1426, "NaN"],
    [20395, "NaN"],
    [1455, "NaN"],
    ["NaN", 5917],
    ["NaN", 2739]
  ]
}
```

5.9.8. Flow Classification API

The *Flow Classification API* can be used to update, create or delete flow classification rules.



If not otherwise specified the **Content-Type** of the response is `application/json`.

GETs (Reading Data)

Resource	Description
<code>/classifications</code>	Retrieve a list of all enabled rules. The request is limited to 10 items by default. Setting limit to 0 returns ALL enabled rules.
<code>/classifications/{id}</code>	Retrieve the rule identified by <code>{id}</code> .
<code>/classifications/groups</code>	Retrieve all existing groups. The request is limited to 10 items by default. Setting limit to 0 returns ALL enabled rules.
<code>/classifications/groups/{id}</code>	Retrieve the group identified by <code>{id}</code> . Supports downloading all rules of the given group in CSV format. For this, set the accept header: Accept: text/comma-separated-values .
<code>/classifications/protocols</code>	Retrieve all supported tcp protocols.

The `/classifications` endpoint supports the following url parameters:



The given filters are combined using a logical **AND**. There is no support for using **OR** logic, or combinations thereof.

name	default
comment	groupFilter
(none)	The group to filter the rules by. Should be the id retrieved by <code>/classifications/groups</code> . It is not possible to retrieve the rules for a disabled group.
query	(none)

Examples

Retrieve all enabled rules

```
curl -X GET -u admin:admin http://localhost:8980/opennms/rest/classifications
```

Response

```
[
  {
    "group": {
      "description": "Classification rules defined by the user",
      "enabled": true,
      "id": 2,
      "name": "user-defined",
      "position": 10,
      "readOnly": false,
      "ruleCount": 1
    },
    "id": 1,
    "ipAddress": null,
    "name": "http",
    "port": "80",
    "position": 0,
    "protocols": [
      "TCP"
    ]
  }
]
```

Retrieve all groups

```
curl -X GET -u admin:admin http://localhost:8980/opennms/rest/classifications/groups
```

Response

```
[
  {
    "description": "Classification rules defined by OpenNMS",
    "enabled": false,
    "id": 1,
    "name": "pre-defined",
    "priority": 0,
    "readOnly": true,
    "ruleCount": 6248
  },
  {
    "description": "Classification rules defined by the user",
    "enabled": true,
    "id": 2,
    "name": "user-defined",
    "priority": 10,
    "readOnly": false,
    "ruleCount": 1
  }
]
```

Retrieve specific rule

```
curl -X GET -u admin:admin http://localhost:8980/opennms/rest/classifications/1
```

Response

```
{
  "group": {
    "description": "Classification rules defined by the user",
    "enabled": true,
    "id": 2,
    "name": "user-defined",
    "priority": 10,
    "readOnly": false,
    "ruleCount": 1
  },
  "id": 1,
  "ipAddress": null,
  "name": "http",
  "port": "80",
  "position": 0,
  "protocols": [
    "TCP"
  ]
}
```

Retrieve specific group

```
curl -X GET -H "Accept: application/json" -u admin:admin  
http://localhost:8980/opennms/rest/classifications/groups/1
```

Response

```
{  
  "description": "Classification rules defined by OpenNMS",  
  "enabled": false,  
  "id": 1,  
  "name": "pre-defined",  
  "priority": 0,  
  "readOnly": true,  
  "ruleCount": 6248  
}
```

Retrieve group as CSV

```
curl -X GET -H "Accept: text/comma-separated-values" -u admin:admin  
http://localhost:8980/opennms/rest/classifications/groups/2
```

Response

```
name;ipAddress;port;protocol  
http;;80;TCP
```

POSTs (Creating Data)

Resource	Description
/classifications	Post a new rule or import rules from CSV. If multiple rules are imported (to user-defined group) from a CSV file all existing rules are deleted.
/classifications/classify	Classify the given request based on all enabled rules.

Examples

Create a single rule

```
curl -X POST -H "Content-Type: application/json" -u admin:admin -d '{"name": "http",  
"port": "80,8080", "protocols": ["tcp", "udp"]}'  
http://localhost:8980/opennms/rest/classifications
```


Response

```
HTTP/1.1 201 Created
Date: Thu, 08 Feb 2018 14:44:27 GMT
Location: http://localhost:8980/opennms/rest/classifications/6616
```

Verify classification engine (rule exists)

```
curl -X POST -H "Content-Type: application/json" -u admin:admin -d '{"protocol":
"tcp", "ipAddress": "192.168.0.1", "port" : "80"}'
http://localhost:8980/opennms/rest/classifications/classify
```

Response

```
{
  "classification": "http"
}
```

Verify classification engine (no rule exists)

```
curl -X POST -H "Content-Type: application/json" -u admin:admin -d '{"protocol":
"tcp", "ipAddress": "192.168.0.1", "port" : "8980"}'
http://localhost:8980/opennms/rest/classifications/classify
```

Response

```
HTTP/1.1 204 No Content
```

Import rules from CSV

```
curl -X POST -H "Content-Type: text/comma-separated-values" -u admin:admin -d
$name;ipAddress;port;protocol\nOpenNMS;;8980;tcp,udp\n'
http://localhost:8980/opennms/rest/classifications/groups/{id}\?hasHeader=true
```

Response

```
HTTP/1.1 204 No Content
```

Import rules from CSV (data with errors)

```
curl -X POST -H "Content-Type: text/comma-separated-values" -u admin:admin -d
$'OpenNMS;;INCORRECT;tcp,udp\nhttp;;80,8080;ULF'
http://localhost:8980/opennms/rest/classifications\?hasHeader=false
```

Response

```
{
  "errors": {
    "1": {
      "context": "port",
      "key": "rule.port.definition.invalid",
      "message": "Please provide a valid port definition. Allowed values are numbers between 0 and 65536. A range can be provided, e.g. \"4000-5000\", multiple values are allowed, e.g. \"80,8080\""
    },
    "2": {
      "context": "protocol",
      "key": "rule.protocol.doesnotexist",
      "message": "The defined protocol 'ULF' does not exist"
    }
  },
  "success": false
}
```

PUTs (Updating Data)

Resource	Description
<code>/classifications/{id}</code>	Update a rule identified by <code>{id}</code> . The id of the rule cannot be changed.
<code>/groups/{id}</code>	Retrieve the rule identified by <code>{id}</code> .
<code>/classifications/groups</code>	Update a group. At the moment, only the enabled property can be changed

DELETEs (Deleting Data)

Resource	Description
<code>/classifications?groupId={groupId}</code>	Deletes all rules of a given group.
<code>/groups/{id}</code>	Delete the given group and all it's containing rules.

5.9.9. Foreign Sources

ReSTful service to the OpenNMS Meridian 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 <i>XML</i> or <i>JSON</i> .)
<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.

5.9.10. Graph API

The *Graph API* can be used to read existing graph and graph containers in the system and is an `api/v2` endpoint.



If not otherwise specified the `Content-Type` of the request and response is `application/json`.

Resource	Method	Description
<code>/graphs</code>	GET	Retrieves the meta data of all registered graph containers.
<code>/graphs/{containerId}</code>	GET	Retrieve the container identified by <code>{containerId}</code> . Returns a 404 if a Graph Container with <code>{containerId}</code> does not exist.
<code>/graphs/{containerId}/{namespace}</code>	GET	Retrieve the graph with namespace <code>{namespace}</code> from the container <code>{containerId}</code> . Returns a 404 if either the <code>{containerId}</code> does not exist or the container is not containing any graph with namespace <code>{namespace}</code> .
<code>/graphs/{containerId}/{namespace}</code>	POST	Retrieve the view of the graph with namespace <code>{namespace}</code> from the container <code>{containerId}</code> . The body must contain the query to create the view, e.g. <code>{ "semanticZoomLevel": 1, "verticesInFocus": []}</code> If no <code>semanticZoomLevel</code> is provided, 1 is assumed. If no <code>verticesInFocus</code> is provided, the default focus of the graph is used. Valid values for <code>verticesInFocus</code> are vertex references (namespace:id tuple, e.g. <code>dummy:1</code>).

At the moment, the context of a search must always be a concrete *namespace* and can never be over all graphs or containers.

Resource	Method	Description
<code>/graphs/search/suggestions/{namespace}?s={search}</code>	GET	Provides suggestions in the context of <code>{namespace}</code> for the input <code>{search}</code> . Each suggestion has the following properties: <code>context</code> : The context of the suggestion, e.g. Node for a node, <code>id</code> : A unique identifier of the representing element of this suggestion, e.g. <code>node id</code> , <code>label</code> : A human readable label, e.g. <code>node label</code> , <code>provider</code> : The source of the suggestion.
<code>/graphs/search/results/{namespace}?providerId={providerId}&criteria={criteria}&context={context}</code>	GET	Resolves a suggestion to actual vertices. The values for <code>{providerId}</code> and <code>{context}</code> are defined in the suggestion. The <code>{criteria}</code> is the <code>{id}</code> property of the suggestion.



The API allows to query any container and graph via ReST, but in most cases you should only query for a graph view.

Examples

Retrieve all graph container infos

Request

```
curl -X GET -u admin:admin http://localhost:8980/opennms/api/v2/graphs
```

Response

```
[
  {
    "description": "Displays all defined applications and their calculated states.",
    "graphs": [
      {
        "description": "Displays all defined applications and their calculated states.",
        "label": "Application Graph",
        "namespace": "application"
      }
    ],
    "id": "application",
    "label": "Application Graph"
  },
  {
    "description": "Displays the hierarchy of the defined Business Services and their computed operational states.",
    "graphs": [
      {
        "description": "Displays the hierarchy of the defined Business Services and their computed operational states.",
```

```

        "label": "Business Service Graph",
        "namespace": "bsm"
    }
],
    "id": "bsm",
    "label": "Business Service Graph"
},
{
    "description": "This Topology Provider displays the topology information
discovered by the Enhanced Linkd daemon. It uses the SNMP information of several
protocols like OSPF, ISIS, LLDP and CDP to generate an overall topology.",
    "graphs": [
        {
            "description": "This Topology Provider displays the topology
information discovered by the Enhanced Linkd daemon. It uses the SNMP information of
several protocols like OSPF, ISIS, LLDP and CDP to generate an overall topology.",
            "label": "Enhanced Linkd Topology Provider",
            "namespace": "nodes"
        }
    ],
    "id": "nodes",
    "label": "Enhanced Linkd Topology Provider"
},
{
    "description": "The VMware Topology Provider displays the infrastructure
information gathered by the VMware Provisioning process.",
    "graphs": [
        {
            "description": "The VMware Topology Provider displays the
infrastructure information gathered by the VMware Provisioning process.",
            "label": "VMware Topology Provider",
            "namespace": "vmware"
        }
    ],
    "id": "vmware",
    "label": "VMware Topology Provider"
}
]

```

Retrieve graph container 'bsm'

Request

```
curl -X GET -u admin:admin http://localhost:8980/opennms/api/v2/graphs/bsm
```

Response

```
{
  "description": "Displays the hierarchy of the defined Business Services and their
computed operational states.",
  "graphs": [
    {
      "defaultFocus": {
        "type": "SELECTION",
        "vertexIds": []
      },
      "description": "Displays the hierarchy of the defined Business Services
and their computed operational states.",
      "edges": [],
      "label": "Business Service Graph",
      "namespace": "bsm",
      "vertices": []
    }
  ],
  "id": "bsm",
  "label": "Business Service Graph"
}
```

Retrieve graph 'layer3-network' in container 'layer3-topology'

Request

```
curl -X GET -u admin:admin http://localhost:8980/opennms/api/v2/graphs/layer3-
topology/layer3-network
```

Response

```
{
  "defaultFocus": {
    "type": "ALL",
    "vertexIds": [
      {
        "id": "horizon",
        "namespace": "layer3-network"
      },
      {
        "id": "database",
        "namespace": "layer3-network"
      },
      {
        "id": "r01",
        "namespace": "layer3-network"
      },
      {
        "id": "r03",
```

```

        "namespace": "layer3-network"
    },
    {
        "id": "r02",
        "namespace": "layer3-network"
    },
    {
        "id": "location-r02",
        "namespace": "layer3-network"
    },
    {
        "id": "net-location",
        "namespace": "layer3-network"
    },
    {
        "id": "location-r03",
        "namespace": "layer3-network"
    },
    {
        "id": "net-central",
        "namespace": "layer3-network"
    },
    {
        "id": "location-r01",
        "namespace": "layer3-network"
    },
    {
        "id": "minion-01",
        "namespace": "layer3-network"
    }
    ]
},
"edges": [
    {
        "id": "layer3-network:database->layer3-network:net-central",
        "namespace": "layer3-network",
        "source": {
            "id": "database",
            "namespace": "layer3-network"
        },
        "target": {
            "id": "net-central",
            "namespace": "layer3-network"
        }
    },
    {
        "id": "layer3-network:horizon->layer3-network:net-central",
        "namespace": "layer3-network",
        "source": {
            "id": "horizon",
            "namespace": "layer3-network"
        }
    }
]

```



```

    },
    "target": {
      "id": "net-central",
      "namespace": "layer3-network"
    }
  },
  {
    "id": "layer3-network:location-r01->layer3-network:net-location",
    "namespace": "layer3-network",
    "source": {
      "id": "location-r01",
      "namespace": "layer3-network"
    },
    "target": {
      "id": "net-location",
      "namespace": "layer3-network"
    }
  },
  {
    "id": "layer3-network:location-r02->layer3-network:net-location",
    "namespace": "layer3-network",
    "source": {
      "id": "location-r02",
      "namespace": "layer3-network"
    },
    "target": {
      "id": "net-location",
      "namespace": "layer3-network"
    }
  },
  {
    "id": "layer3-network:location-r03->layer3-network:net-location",
    "namespace": "layer3-network",
    "source": {
      "id": "location-r03",
      "namespace": "layer3-network"
    },
    "target": {
      "id": "net-location",
      "namespace": "layer3-network"
    }
  },
  {
    "id": "layer3-network:minion-01->layer3-network:net-central",
    "namespace": "layer3-network",
    "source": {
      "id": "minion-01",
      "namespace": "layer3-network"
    },
    "target": {
      "id": "net-central",

```

```

        "namespace": "layer3-network"
    },
    {
        "id": "layer3-network:minion-01->layer3-network:net-location",
        "namespace": "layer3-network",
        "source": {
            "id": "minion-01",
            "namespace": "layer3-network"
        },
        "target": {
            "id": "net-location",
            "namespace": "layer3-network"
        }
    },
    {
        "id": "layer3-network:r01->layer3-network:net-central",
        "namespace": "layer3-network",
        "source": {
            "id": "r01",
            "namespace": "layer3-network"
        },
        "target": {
            "id": "net-central",
            "namespace": "layer3-network"
        }
    },
    {
        "id": "layer3-network:r02->layer3-network:net-central",
        "namespace": "layer3-network",
        "source": {
            "id": "r02",
            "namespace": "layer3-network"
        },
        "target": {
            "id": "net-central",
            "namespace": "layer3-network"
        }
    },
    {
        "id": "layer3-network:r03->layer3-network:net-central",
        "namespace": "layer3-network",
        "source": {
            "id": "r03",
            "namespace": "layer3-network"
        },
        "target": {
            "id": "net-central",
            "namespace": "layer3-network"
        }
    }
}

```

```

],
"enrichment.resolveNodes": "true",
"focus-strategy": "ALL",
"id": "layer3-topology",
"namespace": "layer3-network",
"vertex-status-provider": "true",
"vertices": [
  {
    "foreignID": "database",
    "foreignSource": "opennms-stack",
    "iconKey": "database",
    "id": "database",
    "label": "database",
    "namespace": "layer3-network"
  },
  {
    "foreignID": "horizon",
    "foreignSource": "opennms-stack",
    "iconKey": "opennms_server",
    "id": "horizon",
    "label": "horizon",
    "namespace": "layer3-network"
  },
  {
    "foreignID": "location-r01",
    "foreignSource": "bgp-routers",
    "iconKey": "server",
    "id": "location-r01",
    "label": "location-r01",
    "namespace": "layer3-network"
  },
  {
    "foreignID": "location-r02",
    "foreignSource": "bgp-routers",
    "iconKey": "server",
    "id": "location-r02",
    "label": "location-r02",
    "namespace": "layer3-network"
  },
  {
    "foreignID": "location-r03",
    "foreignSource": "bgp-routers",
    "iconKey": "server",
    "id": "location-r03",
    "label": "location-r03",
    "namespace": "layer3-network"
  },
  {
    "foreignID": "minion-01",
    "foreignSource": "Minions",
    "id": "minion-01",

```

```

    "label": "minion-01",
    "namespace": "layer3-network"
  },
  {
    "iconKey": "vmware-network",
    "id": "net-central",
    "label": "192.168.10.0/24",
    "namespace": "layer3-network"
  },
  {
    "iconKey": "vmware-network",
    "id": "net-location",
    "label": "192.168.50.0/24",
    "namespace": "layer3-network"
  },
  {
    "foreignID": "r01",
    "foreignSource": "bgp-routers",
    "iconKey": "server",
    "id": "r01",
    "label": "r01",
    "namespace": "layer3-network"
  },
  {
    "foreignID": "r02",
    "foreignSource": "bgp-routers",
    "iconKey": "server",
    "id": "r02",
    "label": "r02",
    "namespace": "layer3-network"
  },
  {
    "foreignID": "r03",
    "foreignSource": "bgp-routers",
    "iconKey": "server",
    "id": "r03",
    "label": "r03",
    "namespace": "layer3-network"
  }
]
}

```

Get a view of the graph

Request

```

curl -X POST -u admin:admin -H "Content-Type: application/json" -d '{
  "semanticZoomLevel": 1, "verticesInFocus": ["horizon"] }'
http://localhost:8980/opennms/api/v2/graphs/layer3-topology/layer3-network

```

Response

```
{
  "edges": [
    {
      "id": "layer3-network:horizon->layer3-network:net-central",
      "namespace": "layer3-network",
      "source": {
        "id": "horizon",
        "namespace": "layer3-network"
      },
      "target": {
        "id": "net-central",
        "namespace": "layer3-network"
      }
    }
  ],
  "enrichment.resolveNodes": "true",
  "focus": {
    "semanticZoomLevel": 1,
    "vertices": [
      "horizon"
    ]
  },
  "focus-strategy": "ALL",
  "id": "layer3-topology",
  "namespace": "layer3-network",
  "vertex-status-provider": "true",
  "vertices": [
    {
      "foreignID": "horizon",
      "foreignSource": "opennms-stack",
      "iconKey": "opennms_server",
      "id": "horizon",
      "label": "horizon",
      "namespace": "layer3-network",
      "nodeInfo": {
        "categories": [],
        "foreignId": "horizon",
        "foreignSource": "opennms-stack",
        "id": 90,
        "ipInterfaces": [
          {
            "address": "/192.168.10.4",
            "managed": true,
            "primary": false
          }
        ],
        "label": "horizon",
        "location": "Default"
      }
    }
  ]
}
```

```
    },
    {
      "iconKey": "vmware-network",
      "id": "net-central",
      "label": "192.168.10.0/24",
      "namespace": "layer3-network"
    }
  ]
}
```

Search Suggestions

Request

```
curl -X GET -u admin:admin
http://localhost:8980/opennms/api/v2/graphs/search/suggestions/layer3-
network?s=horizon
```

Response

```
[
  {
    "context": "GenericVertex",
    "id": "horizon",
    "label": "horizon",
    "provider": "LabelSearchProvider"
  },
  {
    "context": "Node",
    "id": "90",
    "label": "horizon",
    "provider": "NodeSearchProvider"
  }
]
```

Search Results

Request

```
curl -X GET -u admin:admin
http://localhost:8980/opennms/api/v2/graphs/search/results/layer3-
network?providerId=NodeSearchProvider&context=Node&criteria=90
```

Response

```
[
  {
    "id": "horizon",
    "namespace": "layer3-network"
  }
]
```

5.9.11. 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.

Resource	Description
/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)

5.9.12. Heatmap

GETs (Reading Data)

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

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

5.9.13. Monitored Services

Obtain or modify the status of a set of monitored services based on a given search criteria, based on nodes, IP interfaces, Categories, or monitored services itself.

Examples:

- `/ifservices?node.label=onms-prd-01`
- `/ifservices?ipInterface.ipAddress=192.168.32.140`
- `/ifservices?category.name=Production`
- `/ifservices?status=A`

GETs (Reading Data)

Resource	Description
<code>/ifservices</code>	Get all configured monitored services for the given search criteria.

Example:

Get the forced unmanaged services for the nodes that belong to the requisition named *Servers*:

```
curl -u admin:admin  
"http://localhost:8980/opennms/rest/ifservices?status=F&node.foreignSource=Servers"
```

PUTs (Modifying Data)

Resource	Description
<code>/ifservices/</code>	Update all configured monitored services for the given search criteria.

Example:

Mark the *ICMP* and *HTTP* services to be forced unmanaged for the nodes that belong to the category *Production*:

```
curl -u admin:admin -X PUT "status=F&services=ICMP,HTTP"  
"http://localhost:8980/opennms/rest/ifservices?category.name=Production"
```

5.9.14. IP Interfaces

Obtain a list of IP interfaces based on a set of criteria.



This API is v2-only, and uses [FIQL](#) for querying.

Examples:

- `/api/v2/ipinterfaces?_s=node.label==onms-prd-01`
- `/api/v2/ipinterfaces?_s=ipAddress==192.168.32.140`

GETs (Reading Data)

Resource	Description
<code>/ipinterfaces</code>	Get all IP interfaces for the given search criteria.

Example:

Get IP interfaces whose node are in the foreign source `Servers`:

```
curl -u admin:admin  
"http://localhost:8980/opennms/api/v2/ipinterfaces?_s=node.foreignSource==Servers"
```

PUTs/POSTs (Modifying Data)

The `/ipinterfaces` ReST API is read-only. If you wish to create new IP interfaces, use the `ipinterfaces` resources under the `nodes endpoint`.

5.9.15. 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](#)

5.9.16. 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> .

5.9.17. Measurements API

The *Measurements API* can be used to retrieve collected values stored in *RRD* (or *JRB*) files and in *Newts*.



Unless specific otherwise, all unit 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 timestamp is relative to the UNIX epoch (January 1st 1970 00:00:00 AM). If < 0, the timestamp is relative to the end option (i.e.: default value is 4 hours ago).
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.
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.
fallback-attribute		Secondary attribute that will be queried in the case the primary attribute does not exist.

Step sizes

The behavior of the **step** parameter changes based on time series strategy that is being used.

When using persistence strategies based on *RRD*, the available step sizes are limited to those defined by the *RRA* when the file was created. The effective step size used will be one that covers the requested period, and is closest to the requested step size. For maximum accuracy, use a step size of 1.

When using *Newts*, the step size can be set arbitrarily since the aggregation is performed at the time of request. In order to help prevent large requests, we limit to the step size of a minimum of 5 minutes, the default collection rate. This value can be decreased by setting the `org.opennms.newts.query.minimum_step` system property.

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>
  <metadata>
    <resources>
      ...
    </resources>
    <nodes>
      ...
    </nodes>
  </metadata>
  <timestamps>1425581100000</timestamps>
  <timestamps>1425581400000</timestamps>
  <timestamps>1425581700000</timestamps>
  ...
</query-response>
```

POSTs (Reading Data)

Resource	Description
<code>/measurements</code>	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[Servers:1424038123222].interfaceSnmp[eth0-
04013f75f101]",
      "transient": "false"
    },
    {
      "aggregation": "AVERAGE",
      "attribute": "ifHCOutOctets",
      "label": "ifHCOutOctets",
      "resourceId": "nodeSource[Servers: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,
        ...
      ]
    }
  ],
  "metadata": {
    "resources": [
      {
        "id": "nodeSource[Servers:1424038123222].interfaceSnmp[eth0-04013f75f101]",
        "label": "eth0-04013f75f101",
        "name": "eth0-04013f75f101",
        "parent-id": "nodeSource[Servers:1424038123222]",
        "node-id": 1
      },
      {
        "id": "nodeSource[Servers:1424038123222].interfaceSnmp[eth0-04013f75f101]",
        "label": "eth0-04013f75f101",
        "name": "eth0-04013f75f101",
        "parent-id": "nodeSource[Servers:1424038123222]",
        "node-id": 1
      }
    ],
    "nodes": [
      {
        "id": 1,
        "label": "Test Server",
        "foreign-source": "Servers",
        "foreign-id": "1424038123222"
      }
    ]
  ]
}

```

```
}  
}
```

More Advanced Expressions

The JEXL 2.1.x library is used to parse the expression string and this also allows java objects and predefined functions to be included in the expression.

JEXL uses a context which is pre-populated by OpenNMS with the results of the query. Several constants and arrays are also predefined as references in the context by OpenNMS.

Constant or prefix	Description
<code>__inf</code>	<code>Double.POSITIVE_INFINITY</code>
<code>__neg_inf</code>	<code>Double.NEGATIVE_INFINITY</code>
<code>NaN</code>	<code>Double.NaN</code>
<code>__E</code>	<code>java.lang.Math.E</code>
<code>__PI</code>	<code>java.lang.Math.PI</code>
<code>__diff_time</code>	Time span between start and end of samples
<code>__step</code>	Difference in time between subsequent values
<code>__i</code>	Index into the samples array which the present calculation is referencing
<code>__AttributeName</code> (where <code>AttributeName</code> is the searched for attribute)	This returns the complete <code>double[]</code> array of samples for <code>AttributeName</code>

OpenNMS predefines a number of functions for use in expressions which are referenced by `namespace:function`. All of these functions return a java double value.

Pre defined functions

Function	Description	Example
jexl:evaluate("_formula"):	Passes a string to the JEXL engine to be evaluated as if it was entered as a normal expression. Like normal expressions, expressions evaluated through this function will return a Java double value. This makes it possible to reference and evaluate a formula which has been stored in OpenNMS as a string variable. The use case for this capability is that it gives us the ability to define and store a per-node and per-value correction formula which can normalise samples from different sample sources.	
math:	References java.lang.Math class	math:cos(20)
strictmath:	References java.lang.StrictMath class	math:cos(20)
fn:	References the class org.opennms.netmgt.measurements.impl.SampleArrayFunctions. This contains several functions which can reference previous samples in the time series.	
fn:arrayNaN("sampleName", n)	References the nth previous sample in the "sampleName" sample series. Replacing the n samples before the start of the series with NaN.	fn:arrayNaN("x", 5)
fn:arrayZero("sampleName", n)	References the nth previous sample in the "sampleName" sample series. Replacing the n samples before the start of the series with 0 (zero).	fn:arrayZero("x", 5)

Function	Description	Example
<code>fn:arrayFirst("sampleName", n)</code>	References the nth previous sample in the "sampleName" sample series. Replacing the n samples before the start of the series with the first sample.	<code>fn:arrayFirst("x", 5)</code>
<code>fn:arrayStart("sampleName", n, constant)</code>	References the nth previous sample in the "sampleName" sample series. Replacing the n samples before the start of the series with a supplied constant.	<code>fn:arrayStart("x", 5, 10)</code>

So for example with these additional variables and functions it is possible to create a Finite Impulse Response (FIR) filter function such as

$$y = a * f(n) + b * f(n-1) + c * f(n-2)$$

using the following expression where a,b and c are string constants and x is a time series value

$$a * x + b * fn:arrayNaN("x", 1) + c * fn:arrayNaN("x", 2)$$

5.9.18. Meta-Data

The actual Meta-Data of nodes, interfaces and services can be queried, added, modified or deleted via ReST endpoints.

GETs (Reading Meta-Data)

Resource	Description
<code>api/v2/nodes/{id}/metadata</code>	Get the list of meta-data associated with this node.
<code>api/v2/nodes/{id}/metadata/{context}</code>	Get the list of meta-data associated with this node filtered by the given context.
<code>api/v2/nodes/{id}/metadata/{context}/{key}</code>	Get the entry for the given context and key associated with this node.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/metadata</code>	Get the list of meta-data associated with this interface.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/metadata/{context}</code>	Get the list of meta-data associated with this interface filtered by the given context.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/metadata/{context}/{key}</code>	Get the entry for the given context and key associated with this interface.

Resource	Description
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/services/{service}/metadata</code>	Get the list of meta-data associated with this service.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/services/{service}/metadata/{context}</code>	Get the list of meta-data associated with this service filtered by the given context.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/services/{service}/metadata/{context}/{key}</code>	Get the entry for the given context and key associated with this service.

POSTs (Adding Meta-Data)

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

Resource	Description
<code>api/v2/nodes/{id}/metadata</code>	Adds a meta-data entry to the given node.
<code>api/v2/nodes/{id}/{ipInterface}/metadata</code>	Adds a meta-data entry to the given interface.
<code>api/v2/nodes/{id}/{ipInterface}/services/{service}/metadata</code>	Adds a meta-data entry to the given service.

PUTs (Modifying Meta-Data)

Resource	Description
<code>api/v2/nodes/{id}/metadata/{context}/{key}/{value}</code>	Sets the given value for the node-level meta-data entry specified by the given context and key.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/metadata/{context}/{key}/{value}</code>	Sets the given value for the interface-level meta-data entry specified by the given context and key.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/services/{service}/metadata/{context}/{key}/{value}</code>	Sets the given value for the service-level meta-data entry specified by the given context and key.

DELETEs (Removing Meta-Data)

Resource	Description
<code>api/v2/nodes/{id}/metadata/{context}</code>	Deletes node-level meta-data with the given context.
<code>api/v2/nodes/{id}/metadata/{context}/{key}</code>	Deletes the node-level meta-data entry for the given context and key.

Resource	Description
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/metadata/{context}</code>	Deletes interface-level meta-data with the given context.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/metadata/{context}/{key}</code>	Deletes the interface-level meta-data entry for the given context and key.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/services/{service}/metadata/{context}</code>	Deletes service-level meta-data with the given context.
<code>api/v2/nodes/{id}/ipinterfaces/{ipInterface}/services/{service}/metadata/{context}/{key}</code>	Deletes the service-level meta-data entry for the given context and key.

5.9.19. 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.
<code>/nodes/{id}/snmpinterfaces</code>	Get the list of SNMP interfaces associated with the given node.
<code>/nodes/{id}/snmpinterfaces/{ifIndex}</code>	Get the specific interface associated with the given node and ifIndex.
<code>/nodes/{id}/categories</code>	Get the list of categories associated with the given node.
<code>/nodes/{id}/categories/{categoryName}</code>	Get the category associated with the given node and category name.
<code>/nodes/{id}/assetRecord</code>	Get the asset record associated with the given node.

POSTs (Adding Data)

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

Resource	Description
<code>/nodes</code>	Add a node.
<code>/nodes/{id}/ipinterfaces</code>	Add an IP interface to the node.
<code>/nodes/{id}/ipinterfaces/{ipAddress}/services</code>	Add a service to the interface for the given node.
<code>/nodes/{id}/snmpinterfaces</code>	Add an SNMP interface to the node.
<code>/nodes/{id}/categories</code>	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
<code>/nodes/{id}</code>	Modify a node with the given ID.
<code>/nodes/{id}/ipinterfaces/{ipAddress}</code>	Modify the IP interface with the given node ID and IP address.
<code>/nodes/{id}/ipinterfaces/{ipAddress}/services/{service}</code>	Modify the service with the given node ID, IP address, and service name.
<code>/nodes/{id}/snmpinterfaces/{ifIndex}</code>	Modify the SNMP interface with the given node ID and ifIndex.
<code>/nodes/{id}/categories/{categoryName}</code>	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.



Deletion of nodes, ipinterfaces and services are asynchronous so they will return 202 (ACCEPTED). Deletion of snmpinterfaces and categories are synchronous calls so they will return 204 (NO_CONTENT) on success.

5.9.20. 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
<code>/notifications</code>	Get a list of notifications.
<code>/notifications/count</code>	Get the number of notifications. (Returns plaintext, rather than <i>XML</i> or <i>JSON</i> .)
<code>/notifications/{id}</code>	Get the notification specified by the given <i>ID</i> .

To acknowledge or unacknowledge a notification, use the `acks` endpoint — see [Acknowledgements](#).

5.9.21. Outage Timelines

GETs (Reading Data)

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

5.9.22. Outages

GETs (Reading Data)

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

5.9.23. Requisitions

RESTful service to the OpenNMS Meridian 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
<code>/requisitions</code>	Get all active requisitions.
<code>/requisitions/count</code>	Get the number of active requisitions. (Returns plaintext, rather than <i>XML</i> or <i>JSON</i> .)
<code>/requisitions/deployed</code>	Get the list of all deployed (active) requisitions.
<code>/requisitions/deployed/count</code>	Get the number of deployed requisitions. (Returns plaintext, rather than <i>XML</i> or <i>JSON</i> .)
<code>/requisitions/{name}</code>	Get the active requisition for the given foreign source name.
<code>/requisitions/{name}/nodes</code>	Get the list of nodes being requisitioned for the given foreign source name.
<code>/requisitions/{name}/nodes/{foreignId}</code>	Get the node with the given foreign <i>ID</i> for the given foreign source name.
<code>/requisitions/{name}/nodes/{foreignId}/interfaces</code>	Get the interfaces for the node with the given foreign <i>ID</i> and foreign source name.
<code>/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}</code>	Get the interface with the given IP for the node with the specified foreign <i>ID</i> and foreign source name.
<code>/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}/services</code>	Get the services for the interface with the specified IP address, foreign <i>ID</i> , and foreign source name.
<code>/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}/services/{service}</code>	Get the given service with the specified IP address, foreign <i>ID</i> , and foreign source name.
<code>/requisitions/{name}/nodes/{foreignId}/categories</code>	Get the categories for the node with the given foreign <i>ID</i> and foreign source name.
<code>/requisitions/{name}/nodes/{foreignId}/categories/{categoryName}</code>	Get the category with the given name for the node with the specified foreign <i>ID</i> and foreign source name.

Resource	Description
<code>/requisitions/{name}/nodes/{foreignId}/assets</code>	Get the assets for the node with the given foreign <i>ID</i> and foreign source name.
<code>/requisitions/{name}/nodes/{foreignId}/assets/{assetName}</code>	Get the value of the asset for the given <i>assetName</i> for the node with the given foreign <i>ID</i> and foreign source name.

POSTs (Adding Data or Updating existing Data)

	Expects JSON/XML
---	------------------

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

PUTs (Modifying Data)

	Expects form-urlencoded
---	-------------------------

Resource	Description
<code>/requisitions/{name}/import</code>	Performs an import/synchronize on the specified foreign source. This turns the "active" requisition into the "deployed" requisition.
<code>/requisitions/{name}/import?rescanExisting=false</code>	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.
<code>/requisitions/{name}</code>	Update the specified foreign source.
<code>/requisitions/{name}/nodes/{foreignId}</code>	Update the specified node for the given foreign source.

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

DELETES (Removing Data)

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

5.9.24. Resources API

The *Resources API* can be used to list or delete resources at the node level and below. This service is especially useful in conjunction with the *Measurements API*.

GETs (Reading Data)

Resource	Description
<code>/resources</code>	Retrieve the full tree of resources in the system (expensive, use with care)
<code>/resources/{resourceId}</code>	Retrieve the tree of resources starting with the named resource ID
<code>/resources/fornode/{nodeCriteria}</code>	Retrieve the tree of resources for a node, given its database ID or <code>foreign-source:foreign-ID</code> tuple

DELETES (Removing Data)

Resource	Description
<code>/resources/{resourceId}</code>	Delete resource with the named resource ID, and all its child resources, if any

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

name	default	comment
depth	varies	GET only. Limits the tree depth for retrieved resources. Defaults to 1 when listing all resources, or to -1 (no limit) when listing a single resource.

Usage examples with curl

Retrieve the tree of resources rooted at the node with database ID 1, by resource ID

```
curl -u admin:admin "http://127.0.0.1:8980/opennms/rest/resources/node%5B1%5D"
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<resource id="node[1]"
  label="anode"
  name="1"
  link="element/node.jsp?node=1"
  typeLabel="Node">
  <children count="11" totalCount="11">
    <resource id="node[1].nodeSnmp[]"
      label="Node-level Performance Data"
      name=""
      typeLabel="SNMP Node Data"
      parentId="node[1]">
      <children/>
      <stringPropertyAttributes/>
      <externalValueAttributes/>
      <rrdGraphAttributes>
        <entry>
          <key>loadavg1</key>
          <value name="loadavg1"
            relativePath="snmp/1"
            rrdFile="loadavg1.jrb"/>
        </entry>
          <key>tcpActiveOpens</key>
          <value name="tcpActiveOpens"
            relativePath="snmp/1"
            rrdFile="tcpActiveOpens.jrb"/>
        </entry>
      </rrdGraphAttributes>
    </resource>
  </children>
</resource>
```

```

<entry>
  <key>memTotalFree</key>
  <value name="memTotalFree"
    relativePath="snmp/1"
    rrdFile="memTotalFree.jrb"/>
</entry>
...
</rrdGraphAttributes>
</resource>
<resource id="node[1].interfaceSnmp[lo]"
  label="lo (10 Mbps)"
  name="lo"
  link="element/snmpinterface.jsp?node=1&index=1"
  typeLabel="SNMP Interface Data"
  parentId="node[1]">
<children/>
<stringPropertyAttributes>
  <entry>
    <key>ifName</key>
    <value>lo</value>
  </entry>
  ...
</stringPropertyAttributes>
<externalValueAttributes>
  <entry>
    <key>ifSpeed</key>
    <value>10000000</value>
  </entry>
  <entry>
    <key>ifSpeedFriendly</key>
    <value>10 Mbps</value>
  </entry>
</externalValueAttributes>
<rrdGraphAttributes>
  ...
  <entry>
    <key>ifHCInOctets</key>
    <value name="ifHCInOctets"
      relativePath="snmp/1/lo"
      rrdFile="ifHCInOctets.jrb"/>
  </entry>
  <entry>
    <key>ifHCOutOctets</key>
    <value name="ifHCOutOctets"
      relativePath="snmp/1/lo"
      rrdFile="ifHCOutOctets.jrb"/>
  </entry>
  ...
</rrdGraphAttributes>
</resource>
...

```

```

</children>
<stringPropertyAttributes/>
<externalValueAttributes/>
<rrdGraphAttributes/>
</resource>

```

Retrieve the tree of resources rooted at the node with database ID 1, without having to construct a resource ID

```
curl -u admin:admin "http://127.0.0.1:8980/opennms/rest/resources/fornode/1"
```

Retrieve the tree of resources rooted at the node with foreign-ID node42 in requisition Servers, by resource ID

```
curl -u admin:admin
"http://127.0.0.1:8980/opennms/rest/resources/nodeSource%5BServers:node42%5D"
```

Retrieve the tree of resources rooted at the node with foreign-ID node42 in requisition Servers, without having to construct a resource ID

```
curl -u admin:admin
"http://127.0.0.1:8980/opennms/rest/resources/fornode/Servers:node42"
```

5.9.25. Realtime Console data

The *Realtime Console (RTC)* calculates the availability for monitored services. Data provided from the RTC is available to the ReST API.

GETs (Reading Data)

Resource	Description
/availability/categories/{category}	Get all nodes and availability data from a given SLA category filter, i.e. Web Servers (Web+Servers)
/availability/categories/{category}/nodes	Get node availability data for each node of a given SLA category filter
/availability/categories/{category}/nodes/{nodeId}	Get detailed service availability for a given node in a given SLA category filter
/availability/nodes/{nodeId}	Get detailed availability for all services on a given node

Example

```

curl -u demo:demo
http://demo.opennms.org/opennms/rest/availability/categories/Web+Servers
curl -u demo:demo http://demo.opennms.org/opennms/rest/availability/categories/nodes
curl -u demo:demo
http://demo.opennms.org/opennms/rest/availability/categories/nodes/31
curl -u demo:demo http://demo.opennms.org/opennms/rest/availability/nodes/31

```

5.9.26. Scheduled Outages

GETs (Reading Data)

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

POSTs (Setting Data)

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

PUTs (Modifying Data)

Parameter	Description
<code>/scheduled-outages/{outageName}/collectd/{package}</code>	to add a specific outage to a collectd's package.
<code>/scheduled-outages/{outageName}/pollerd/{package}</code>	to add a specific outage to a pollerd's package.

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

DELETES (Removing Data)

Parameter	Description
<code>/scheduled-outages/{outageName}</code>	to delete a specific outage.
<code>/scheduled-outages/{outageName}/collected/{package}</code>	to remove a specific outage from a collectd's package.
<code>/scheduled-outages/{outageName}/pollerd/{package}</code>	to remove a specific outage from a pollerd's package.
<code>/scheduled-outages/{outageName}/threshd/{package}</code>	to remove a specific outage from a threshd's package.
<code>/scheduled-outages/{outageName}/notifd</code>	to remove a specific outage from the notifications.

5.9.27. 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
<code>/snmpConfig/{ipAddress}</code>	Get the SNMP configuration for a given IP address.
<code>/snmpConfig/{ipAddress}?location={location}</code>	Get the SNMP configuration for a given IP address at a given location.

PUTs (Modifying Data)

Parameter	Description
<code>/snmpConfig/{ipAddress}</code>	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 Meridian 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
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 *Snmplib* 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 Meridian 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 *Snmplib* 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	attribute snmp-config.xml
snmp-info.xml	
default	restricted to version
restriction	version

attribute snmp-info.xml	attribute snmp-config.xml
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

attribute snmp-info.xml	attribute snmp-config.xml
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
<p>Integer value, which can be null, 1, 2, or 3.</p> <ul style="list-style-type: none"> 1 means noAuth NoPriv 2 means authNo Priv3 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 security Level is 1if a authPassPhrase and no 	<p>authPassPhrase</p>
96 privPas	

attribute snmp-info.xml	attribute snmp-config.xml
auth-passphrase	Op3nNMSv3
v3	
authProtocol	auth-protocol
MD5	v3
only MD5, SHA, SHA-224, SHA-256, SHA-512 are valid arguments	privPassPhrase
privacy-passphrase	Op3nNMSv3
v3	
privProtocol	privacy-protocol
DES	v3
only DES, AES, AES192 or AES256 are valid arguments.	engineId

attribute	snmp-config.xml
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`.

5.9.28. SNMP Interfaces

Obtain a list of SNMP interfaces based on a set of criteria.



This API is v2-only, and uses [FIQL](#) for querying.

Examples:

- `/api/v2/snmpinterfaces?_s=node.label==onms-prd-01`
- `/api/v2/snmpinterfaces?_s=ifIndex==6`

GETs (Reading Data)

Resource	Description
<code>/snmpinterfaces</code>	Get all SNMP interfaces for the given search criteria.

Example:

Get SNMP interfaces whose node are in the foreign source `Servers` and are associated with IP address `127.0.0.1`:

```
curl -u admin:admin
"http://localhost:8980/opennms/api/v2/snmpinterfaces?_s=node.foreignSource==Servers;ip
Interfaces.ipAddress=127.0.0.1"
```

PUTs/POSTs (Modifying Data)

The `/snmpinterfaces` ReST API is read-only. If you wish to create new SNMP interfaces, use the `snmpinterfaces` resources under the `nodes` endpoint.

5.9.29. 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.



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)

Param eter	Description
<code>/users</code>	Get a list of users.
<code>/users/{username}</code>	Get a specific user, by username.

POSTs (Adding Data)

Parameter	Description
<code>/users</code>	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 <i>MD5</i> based algorithm, you need to pass an element named <code>passwordSalt</code> with text <code>true</code> after the password element (or key/value pairs if using <i>JSON</i>). Note: You may add the query parameter <code>hashPassword=true</code> to tell OpenNMS you are passing an unencrypted password; it will hash and salt the password when it is saved.

PUTs (Modifying Data)

Parameter	Description
<code>/users/{username}</code>	Update an existing user's full-name, user-comments, password, passwordSalt and duty-schedule values. Note: If you are setting the password, you may also add the query parameter <code>hashPassword=true</code> to tell OpenNMS you are passing an unencrypted password; it will hash and salt the password when it is saved.
<code>/users/{username}/roles/{rolename}</code>	Add a security role to the user. (new in OpenNMS 19)

DELETES (Removing Data)

Resource	Description
<code>/users/{username}</code>	Delete a user.
<code>/users/{username}/roles/{rolename}</code>	Remove a security role from the user. (new in OpenNMS 19)

5.9.30. User Defined Links

User defined links (UDLs) for the Enlinkd topology can be queried, added, modified or deleted via ReST endpoints.

GETs (Reading UDLs)

Resource	Description
<code>api/v2/userdefinedlinks</code>	Get the list of UDLs.
<code>api/v2/userdefinedlinks/{id}</code>	Retrieve a specific UDL by database ID.

POSTs (Creating UDLs)

Resource	Description
<code>api/v2/userdefinedlinks</code>	Add a new UDL.

Usage examples with curl

Create a new UDL between node with ID 1 and node with ID 2.

```
curl -v -X POST -u admin:admin -H "Content-Type: application/json" -d '{"node-id-a": 1, "node-id-z": 2, "component-label-a": "tp1", "component-label-z": "tp2", "link-id": "n1:tp1->n2:tp2", "owner": "me"}'
http://localhost:8980/opennms/api/v2/userdefinedlinks
```

DELETES (Removing UDLs)

Resource	Description
<code>api/v2/userdefinedlinks/{id}</code>	Delete a UDL by database ID.

5.9.31. SNMP Trap Northbender Interface Configuration

GETs (Reading Data)

Resource	Description
<code>/config/snmptrap-nbi</code>	Gets full content of the configuration.
<code>/config/snmptrap-nbi/status</code>	Gets the status of the SNMP Trap NBI (returns either true or false).
<code>/config/snmptrap-nbi/destinations</code>	Gets the name of all the existing destinations.
<code>/config/snmptrap-nbi/destinations/{name}</code>	Gets the content of the destination named {name}

PUTs (Update defaults)

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

Resource	Description
<code>/config/snmptrap-nbi/status?enabled=(true;false)</code>	Sets the status of the SNMP Trap NBI.

POSTs (Adding Data)

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

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

Resource	Description
<code>/config/snmptrap-nbi</code>	Updates the full content of the configuration.
<code>/config/snmptrap-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 SNMP Trap NBI will be notified about the configuration change.

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

DELETES (Remove Data)

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

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

5.9.32. Email Northbound Interface Configuration

GETs (Reading Data)

Resource	Description
<code>/config/email-nbi</code>	Gets full content of the configuration.
<code>/config/email-nbi/status</code>	Gets the status of the Email NBI (returns either true or false).
<code>/config/email-nbi/destinations</code>	Gets the name of all the existing destinations.
<code>/config/email-nbi/destinations/{name}</code>	Gets the content of the destination named {name}

PUTs (Update defaults)

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

Resource	Description
<code>/config/email-nbi/status?enabled=(true;false)</code>	Sets the status of the Email NBI.

POSTs (Adding Data)

POST 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</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}

5.9.33. 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}

Resource	Description
<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.

PUTs (Update defaults)

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

Resource	Description
<code>config/javamail/default/readmail/{name}</code>	Sets the readmail named {name} as the new default.
<code>config/javamail/default/sendmail/{name}</code>	Sets the sendmail named {name} as the new default.

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/javamail/readmails/{name}</code>	Updates the content of the readmail configuration named {name}
<code>/config/javamail/sendmails/{name}</code>	Updates the content of the sendmail configuration named {name}
<code>/config/javamail/end2ends/{name}</code>	Updates the content of the end2end mail configuration named {name}

DELETES (Remove Data)

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

Resource	Description
<code>/config/javamail/readmails/{name}</code>	Removes the readmail configuration named {name}
<code>/config/javamail/sendmails/{name}</code>	Removes the sendmail configuration named {name}
<code>/config/javamail/end2ends/{name}</code>	Removes the end2end mail configuration named {name}

5.9.34. Syslog Northbound Interface Configuration

GETs (Reading Data)

Resource	Description
<code>/config/syslog-nbi</code>	Gets full content of the configuration.
<code>/config/syslog-nbi/status</code>	Gets the status of the Syslog NBI (returns either true or false).
<code>/config/syslog-nbi/destinations</code>	Gets the name of all the existing destinations.
<code>/config/syslog-nbi/destinations/{name}</code>	Gets the content of the destination named {name}

PUTs (Update defaults)

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

Resource	Description
<code>/config/syslog-nbi/status?enabled=(true;false)</code>	Sets the status of the Syslog NBI.

POSTs (Adding Data)

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

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

Resource	Description
<code>/config/syslog-nbi</code>	Updates the full content of the configuration.
<code>/config/syslog-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 Syslog NBI will be notified about the configuration change.

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

DELETES (Remove Data)

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

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

5.9.35. Business Service Monitoring

Every aspect of the *Business Service Monitoring* feature can be controlled via a ReST API. The API's endpoint for managing *Business Services* is located at `/opennms/api/v2/business-services`. It supports *XML* content to represent the *Business Services*. The schema file describing the API model is located in `$OPENNMS_HOME/share/xsds/business-service-dto.xsd`. The responses generated by the ReST API do also include `location` elements that contain references to other entities managed by the API. The *Business Service* response data model for the ReST API has the following basic structure:

Sample Business Service details response XML representation

```
<business-service>
  <id>42</id>
  <name>Datacenter North</name>
  <attributes/>
  <ip-service-edges>
    <ip-service-edge>
      <id>23</id>
      <operational-status>WARNING</operational-status>
      <map-function>
        <type>Identity</type>
      </map-function>
      <location>/api/v2/business-services/2/edges/23</location>
      <reduction-keys>
        <reduction-key>
uei.opennms.org/nodes/nodeLostService::12:10.10.10.42:ICMP</reductionKey>
          <reduction-key>uei.opennms.org/nodes/nodeDown::12</reductionKey>
        </reduction-keys>
        <weight>1</weight>
      </ip-service-edge>
    </ip-service-edges>
  <reduction-key-edges>
    <reduction-key-edge>
      <id>111</id>
      <operational-status>INDETERMINATE</operational-status>
      <map-function>
        <type>Identity</type>
      </map-function>
    </reduction-key-edge>
  </reduction-key-edges>
</business-service>
```



```

    </map-function>
    <location>/api/v2/business-services/42/edges/111</location>
    <reduction-keys>
      <reduction-key>my-reduction-key1</reduction-key>
    </reduction-keys>
    <weight>1</weight>
  </reduction-key-edge>
</reduction-key-edges>
<child-edges>
  <child-edge>
    <id>123</id>
    <operational-status>MINOR</operational-status>
    <map-function>
      <type>Identity</type>
    </map-function>
    <location>/api/v2/business-services/42/edges/123</location>
    <reduction-keys/>
    <weight>1</weight>
    <child-id>43</child-id>
  </child-edge>
</child-edges>
<parent-services><parent-service>144</parent-service></parent-services>
<reduce-function><type>HighestSeverity</type></reduce-function>
<operational-status>INDETERMINATE</operational-status>
<location>/api/v2/business-services/146</location>
</business-service>

```

Sample Business Service creation request XML representation

```
<business-service>
  <name>Datacenter North</name>
  <attributes/>
  <ip-service-edges>
    <ip-service-edge>
      <ip-service-id>99</ip-service-id>
      <map-function>
        <type>Identity</type>
      </map-function>
      <weight>1</weight>
    </ip-service-edge>
  </ip-service-edges>
  <reduction-key-edges>
    <reduction-key-edge>
      <reduction-key>my-reduction-key1</reduction-key>
      <map-function>
        <type>Identity</type>
      </map-function>
      <weight>1</weight>
    </reduction-key-edge>
  </reduction-key-edges>
  <child-edges>
    <child-edge>
      <child-id>43</child-id>
      <map-function>
        <type>Identity</type>
      </map-function>
      <weight>1</weight>
    </child-edge>
  </child-edges>
  <reduce-function><type>HighestSeverity</type></reduce-function>
</business-service>
```

The whole model is defined in `jetty-webapps/opennms/WEB-INF/lib/org.opennms.features.bsm.rest.api-*.jar` which can be used as a dependency for a Java program to query the API.

GETs (Reading Data)

Resource	Description
<code>/opennms/api/v2/business-services</code>	Provides a brief list of all defined <i>Business Services</i>
<code>/opennms/api/v2/business-services/{id}</code>	Returns the <i>Business Service</i> identified by <code>id</code> included the current operational state
<code>/opennms/api/v2/business-services/edges/{edgeId}</code>	Returns the edge of the <i>Business Service</i> identified by <code>edgeId</code>

Resource	Description
<code>/opennms/api/v2/business-services/functions/map</code>	Provides a list of supported <i>Map Function</i> definitions
<code>/opennms/api/v2/business-services/functions/map/{name}</code>	Returns the definition of the <i>Map Function</i> identified by <code>name</code>
<code>/opennms/api/v2/business-services/functions/reduce/</code>	Provides a list of supported <i>Reduce Function</i> definitions
<code>/opennms/api/v2/business-services/functions/reduce/{name}</code>	Returns the definition of the <i>Reduce Function</i> identified by <code>name</code>

PUTs (Modifying Data)

Resource	Description
<code>/opennms/api/v2/business-services/{id}</code>	Modifies an existing <i>Business Service</i> identified by <code>id</code>

POSTs (Adding Data)

Resource	Description
<code>/opennms/api/v2/business-services</code>	Creates a new <i>Business Service</i>
<code>/opennms/api/v2/business-services/{id}/ip-service-edge</code>	Adds an edge of type <i>IP Service</i> to the <i>Business Service</i> identified by <code>id</code>
<code>/opennms/api/v2/business-services/{id}/reduction-key-edge</code>	Adds an edge of type <i>Reduction Key</i> to the <i>Business Service</i> identified by <code>id</code>
<code>/opennms/api/v2/business-services/{id}/child-edge</code>	Adds an edge of type <i>Business Service</i> to the <i>Business Service</i> identified by <code>id</code>
<code>/opennms/api/v2/daemon/reload</code>	Reload the configuration of the <i>Business Service Daemon</i>

DELETES (Removing Data)

Resource	Description
<code>/opennms/api/v2/business-services/{id}</code>	Deletes the <i>Business Service</i> identified by <code>id</code>
<code>/opennms/api/v2/business-services/{id}/edges/{edgeId}</code>	Removes an edge with the identifier <code>edgeId</code> from the <i>Business Service</i> identified by <code>id</code>

5.9.36. Discovery

This endpoint can be used to trigger a one-time discovery scan.

POSTs (Submitting one-time scan configuration)

Resource	Description
/opennms/api/v2/discovery	Submits an one-time scan configuration

The following XML structure is used to define a scan job.

Sample configuration file `discovery.xml`

```
<discoveryConfiguration>
  <specifics>
    <specific>
      <location>Default</location>
      <retries>3</retries>
      <timeout>2000</timeout>
      <foreignSource>My-ForeignSource</foreignSource>
      <content>192.0.2.1</content>
    </specific>
  </specifics>
  <includeRanges>
    <includeRange>
      <location>Default</location>
      <retries>3</retries>
      <timeout>2000</timeout>
      <foreignSource>My-ForeignSource</foreignSource>
      <begin>192.0.2.10</begin>
      <end>192.0.2.254</end>
    </includeRange>
  </includeRanges>
  <excludeRanges>
    <excludeRange>
      <begin>192.0.2.60</begin>
      <end>192.0.2.65</end>
    </excludeRange>
  </excludeRanges>
  <includeUrls>
    <includeUrl>
      <location>Default</location>
      <retries>3</retries>
      <timeout>2000</timeout>
      <foreignSource>My-ForeignSource</foreignSource>
      <content>http://192.0.2.254/addresses.txt</content>
    </includeUrl>
  </includeUrls>
</discoveryConfiguration>
```

The scan itself can be triggered by posting the configuration to the ReST endpoint as follows:

Trigger one-time scan

```
curl -H "Content-Type: application/xml" -u admin:admin -X POST -d @discovery.xml  
http://localhost:8980/opennms/api/v2/discovery
```

5.9.37. Situation Feedback

Situation Feedback is an optional feature that allows for the management of User Contributed Feedback regarding Correlation of Alarms.

It is a simple API that allows for the submission and retrieval of Correlation Feedback.

GETs (Reading Data)

Parameter	Description
<code>/situation-feedback/{situationId}</code>	Get Situation Feedback for the Situation with AlarmId = {situationId}.



situationId is the same as the *AlarmId* for the Situation.

Usage examples with curl

Retrieve Situation Feedback for the Situation with AlarmId == 210

```
curl -u admin:admin "http://localhost:8980/opennms/rest/situation-feedback/210"
```

Response

```
[
  {
    "situationKey": "uei.opennms.org/alarms/trigger:localhost:0.0.0.0:FEEDBACK_A",
    "situationFingerprint": "NDg3ZjdiMjJmNjgzMTJkMmMxYmJjOTNiMWFLYTQ0NWI=",
    "alarmKey": "uei.opennms.org/alarms/trigger:localhost:0.0.0.0:ALARM_A",
    "feedbackType": "CORRECT",
    "reason": "okay",
    "user": "admin",
    "timestamp": 1535057676107
  },
  {
    "situationKey": "uei.opennms.org/alarms/trigger:localhost:0.0.0.0:FEEDBACK_A",
    "situationFingerprint": "NDg3ZjdiMjJmNjgzMTJkMmMxYmJjOTNiMWFLYTQ0NWI=",
    "alarmKey": "uei.opennms.org/alarms/trigger:localhost:0.0.0.0:ALARM_B",
    "feedbackType": "CORRECT",
    "reason": "okay",
    "user": "admin",
    "timestamp": 1535057676107
  }
]
```

POSTs (Adding Data)

POST requires using application/json as a Content-Type.

On a successful request, the Feedback will be persisted

Resource	Description
<code>/situation-feedback/{situationId}</code>	Submits feedback for the Situation identified with AlarmId = {situationId}.

Usage examples with curl

Submit positive Feedback for a Situation with 2 alarms.

```
curl -X POST -H "Accept: application/json" -H "Content-Type: application/json" -u
admin:admin -d @feedback.json http://127.0.0.1:8980/opennms/rest/situation-
feedback/210
```

```
[
  {
    "situationKey": "uei.opennms.org/alarms/trigger:localhost:0.0.0.0:FEEDBACK_A",
    "situationFingerprint": "NDg3ZjdiMjJmNjgzMTJkMmMxYmJjOTNiMWFLYTQ0NWI=",
    "alarmKey": "uei.opennms.org/alarms/trigger:localhost:0.0.0.0:ALARM_A",
    "feedbackType": "CORRECT",
    "reason": "okay",
    "user": "admin",
    "timestamp": 1535057676107
  },
  {
    "situationKey": "uei.opennms.org/alarms/trigger:localhost:0.0.0.0:FEEDBACK_A",
    "situationFingerprint": "NDg3ZjdiMjJmNjgzMTJkMmMxYmJjOTNiMWFLYTQ0NWI=",
    "alarmKey": "uei.opennms.org/alarms/trigger:localhost:0.0.0.0:ALARM_B",
    "feedbackType": "CORRECT",
    "reason": "okay",
    "user": "admin",
    "timestamp": 1535057676107
  }
]
```

Response

204 No Content

5.10. ReST API Examples

5.10.1. 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";
    }
}

```

5.10.2. 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.

5.10.3. 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
```

5.10.4. 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

```

Chapter 6. Develop Documentation

This document is the guideline for people who wish to contribute to writing documentation for the OpenNMS project. The OpenNMS software is free and open source, contribution of any kind is welcome. We ask that you observe the rules and guidelines outlined here to maintain consistency across the project.

Each (sub)project is represented as a section of the documentation. Each section will produce a HTML output in the file system that is generated in the `target/generated` sources folder.

The chosen file format for documentation is AsciiDoc ([AsciiDoc Homepage](#)). Document files use the `.adoc` file extension.

Note that there are different ways to contribute documentation, each suitable for the different use cases:

- Tutorials and How To's should be published on the [OpenNMS Wiki](#). For example: you want to describe how to use the Net-SNMP agent and the SNMP monitor from OpenNMS to solve a special use case with OpenNMS.
- The documentation in the source code should be formal technical documentation. The writing style should be accurate and concise. However, ensure that you explain concepts in detail and do not make omissions.

6.1. File Structure in opennms-doc

Directory	Contents
<code>guide-admin/</code>	module with the guide for administrators configuring, optimizing and running OpenNMS
<code>guide-development/</code>	module with the guide for those who want to develop OpenNMS
<code>guide-install/</code>	module with the guide of how to install OpenNMS on different operating systems
<code>releasenotes/</code>	module with the changelog and release notes

6.2. Writing

The following rules will help you to commit correctly formatted and prepared documentation for inclusion in the OpenNMS project. It is important that we maintain a level of consistency across all of our committers and the documentation they produce.

When writing place a single sentence on each line. This makes it easy to move content around, and also easy to spot long, or fragmented, sentences. This will also allow us to assign comments on a sentence in GitHub which will facilitate easier merging.



Other than writing documentation, you can help out by providing comments on documentation, reviewing, suggesting improvements or reporting bugs. To do this head over to [GitHub](#).

6.2.1. Conventions for text formatting

The following conventions are used:

- File names and path are written in ``poller-configuration.xml`` they will be rendered in: `poller-configuration.xml`;
- Names that indicate special attention, e.g., this configuration matches `*any*` entry: this is rendered as: this configuration matches **any** entry;
- `_Italics_` is rendered as *Italics* and used for emphasis and indicates internal names and abbreviations;
- `*Bold*` is rendered as **Bold** and should be used sparingly, for strong emphasis only;
- `+methodName()+` is rendered as `methodName()` and is also used for literals, (note: the content between the `+` signs *will* be parsed);
- ``command`` is rendered as `command` (typically used for command-line or parts used in configuration files) (note: the content between the ``` signs *will not* be parsed);
- ``my/path/`` is rendered as `my/path/`. This is used for file names and paths;
- ```double quote"` (which is two grave accents to the left and two acute accents to the right) renders as ```double quote"`;
- ``single quote'` (which is a single grave accent to the left and a single acute accent to the right) renders as ``single quote'`.

6.2.2. Gotchas

- Always leave a blank line at the top of the documents section. It might be the title ends up in the last paragraph of the document.
- Start in line 2 setting a relative path to the images directory to picture rendering on GitHub:

```
// Allow image rendering
:imagesdir: relative/path/to/images/dir
```

- Always leave a blank line at the end of documents.
- As `{}` are used for AsciiDoc attributes, everything inside will be treated as an attribute. To avoid this you have to escape the opening brace: `\\{`. If you do not escape the opening brace, the braces and the text inside them will be removed without any warning.
- Force line breaks with `` + `` at the end of the line followed by a line break.

Example in source: force line break

```
This is the first line +  
and this a forced 2nd line
```

Rendered output with forced line break

This is the first line
and this a forced 2nd line

6.3. Headings and document structure

Each document starts with headings from level zero (the document title). Each document should have an ID. In some cases sections in the document also need to have ID's; this depends on where they fit in the overall structure. If you wish to have a link to specific content that content has to have an ID. A missing ID in a mandatory place will cause the build to 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, use the `leveloffset` attribute.

Subsequent headings in a document should use the following syntax:

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

6.4. Links

Use the target ID to link to other parts of the manual. To use a target ID, use this syntax:

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

This will render as: [Links](#)



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, use this syntax:

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



Having a lot of linked text may work well in a web context but is distracting in print. The documentation is intended for both media so be considerate of this in your usage.

To use an external link, use this syntax:

```
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:

```
http://www.opennms.org/
```

Which renders as: <http://www.opennms.org/>



It is acceptable to have a period trailing after the URL, it will not render as a part of the link.

6.5. Admonitions and useful notes

Use to define specific sections, such as Notes, Tips and Important information. We encourage their use in the documentation as long as they are used appropriately. Choose from the following:

Source template for making a note for additional hints

```
NOTE: This is my note.
```

This is how its rendered:



This is my note.

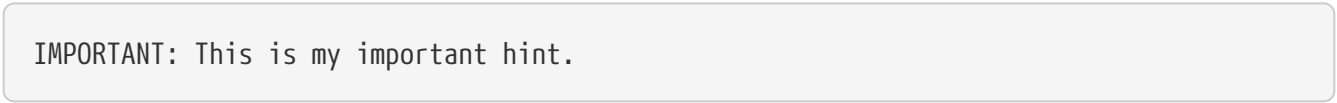
Source for giving a tip

```
TIP: This is my tip.
```

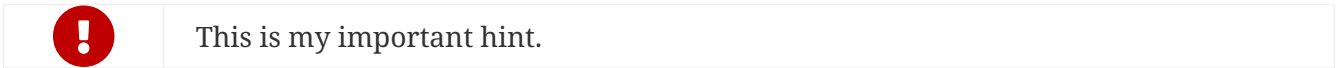
This is how its rendered:



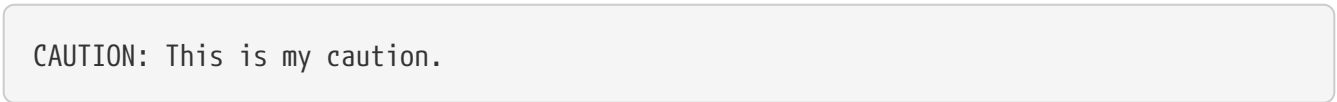
Source for giving a important hint



This is how its rendered:



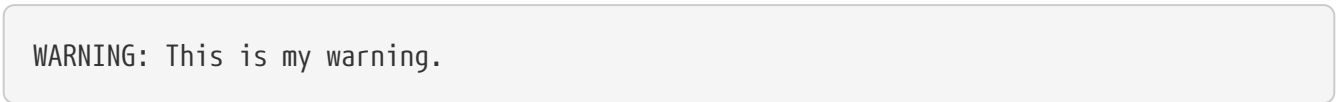
Source for giving a caution



This is how its rendered:



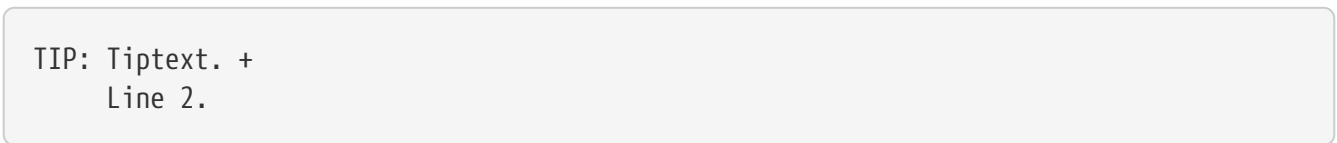
Source for giving a warning



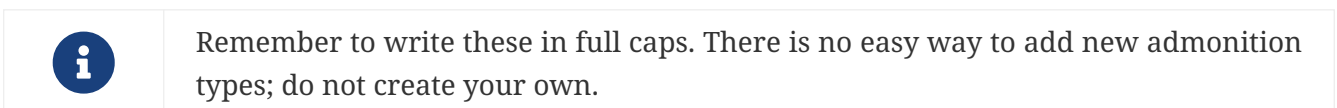
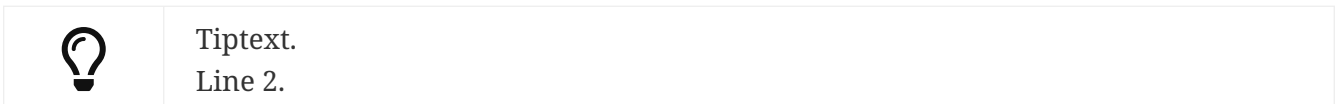
This is how its rendered:



A multiline variation:



Which is rendered as:



6.6. Attributes

Common attributes you can use in documents:

- {opennms-version} - rendered as "2020.1.17"

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 you can use:

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

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

6.8. Tables

Use tables to represent structured information. A table is constructed in the following manner:

```
[options="header, autowidth"]  
|===  
| Parameter      | Description                | Required | Default value  
| `myFirstParm`  | my first long description  | required | `myDefault`  
| `myScndParm`  | my second long description | required | `myDefault`  

```

This is rendered as:

Parameter	Description	Required	Default value
myFirstParm	my first long description	required	myDefault
myScndParm	my second long description	required	myDefault



Please align your columns in the AsciiDoc source for better readability when editing in text view. If you have a very long description, break at 120 characters and align the text to improve source readability.

```
[options="header, autowidth"]
|====
| Parameter | Description | Required | Default value
| `basic-authentication` | Authentication credentials to perform basic authentication. Credentials should comply to http://www.rfc-editor.org/rfc/rfc1945.txt[RFC1945] section 11.1, without the Base64 encoding part. That's: be a string made of the concatenation of: +
| 1- the user ID; +
| 2- a colon; +
| 3- the password. +
| `basic-authentication` takes precedence over the `user` and `password` parameters. | optional | '-'
| `header[0-9]+` | Additional headers to be sent along with the request. Example of valid parameter's names are `header0`, `header1` and `header180`. `header` is *not* a valid parameter name. | optional | '-'
|====
```

Figure 1. Example in AsciiDoc source for very long table descriptions

this is rendered as:

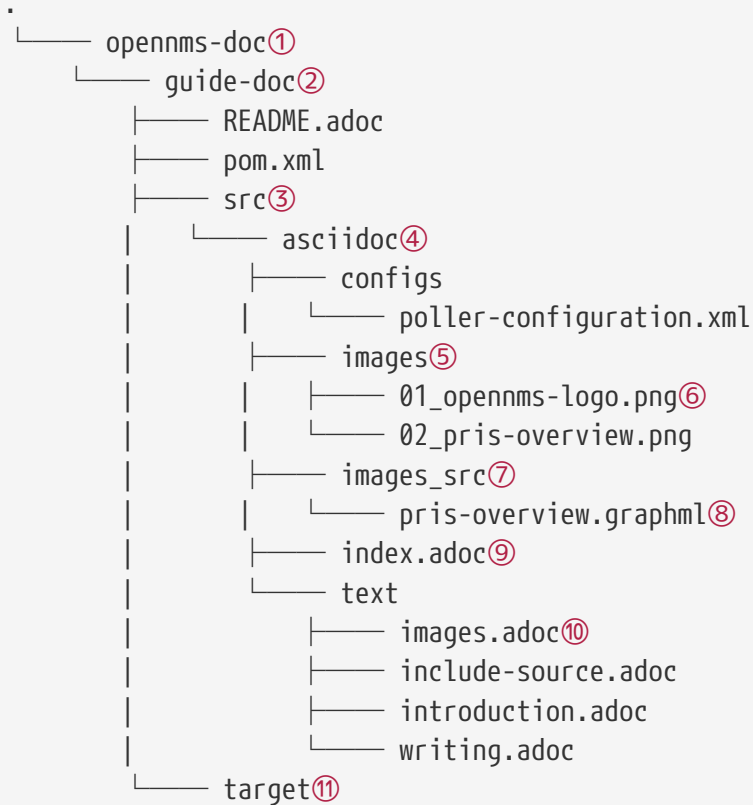
Parameter	Description	Required	Default value
basic-authentication	Authentication credentials to perform basic authentication. Credentials should comply to RFC1945 section 11.1, without the Base64 encoding part. That's: be a string made of the concatenation of: 1- the user ID; 2- a colon; 3- the password. basic-authentication takes precedence over the user and password parameters.	optional	-
header[0-9]+	Additional headers to be sent along with the request. Example of valid parameter's names are header0, header1 and header180. header is not a valid parameter name.	optional	-

6.9. Include images

When visualizing complex problems use images to help explain and provide greater information. We use two directories for images.

The image folder structure mirrors the text structure. This makes it easier to locate the AsciiDoc text file where the image is included.

Example folder structure for image files



- ① This folder contains all documentation modules.
- ② The module for this documentation for target group of documentation contributors.
- ③ Indicates a source folder.
- ④ The documentation root folder.
- ⑤ Folder for images. Images should be *.png or *.jpg if included in the documentation.
- ⑥ The image used; the format is a leading `<number>_` followed by a name using no spaces.
- ⑦ Some images are created from tools like *yED*. This folder should contain the editable version of the file with the same file name.
- ⑧ Editable version of the image source file. Note no spaces in the name.
- ⑨ Main document file which includes all documentation parts and is rendered as `index.html` for the web.
- ⑩ AsciiDoc source file which can include images.
- ⑪ Target folder with generated HTML output after `mvn clean package` has been performed.



All images in the entire manual share the same namespace, it is therefore best practice to use unique identifiers for images.

To include an image file, make sure that it resides in the 'images/' directory relative to the document you're including it in. Then use the following syntax for inclusion in the document:

First included image

```
.This is a caption of the image  
image::docs/02_opennms-logo.png[]
```

Which is rendered as:



Figure 2. This is a caption of the image



The image path for the images you include is relative to the *.adoc source file, where you use the image.

6.10. Code Snippets

You can include code snippets, configuration- or source code files in the documentation. You can enable syntax highlighting by providing the given language parameter. This will work on source code or configuration.

6.10.1. Explicitly defined in the document



Be careful to use this kind of code snippet as sparsely as possible. Code becomes obsolete very quickly; archaic usage practices are detrimental.

If you do wish to include snippets, use the following method:

This is a sample configuration explicitly in the documentation

```
<service name="DNS" interval="300000" user-defined="false" status="on">  
  <parameter key="retry" value="2" />  
  <parameter key="timeout" value="5000" />  
  <parameter key="port" value="53" />  
  <parameter key="lookup" value="localhost" />  
  <parameter key="fatal-response-codes" value="2,3,5" /><!-- ServFail, NXDomain,  
Refused -->  
  <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response" />  
  <parameter key="rrd-base-name" value="dns" />  
  <parameter key="ds-name" value="dns" />  
</service>
```

If there's no suitable syntax highlighter for the code used just omit the language: [source].

Currently the following syntax highlighters are enabled:

- Bash
- Groovy

- Java
- JavaScript
- Python
- XML

For other highlighters that could be added see <https://code.google.com/p/google-code-prettify/>.

6.10.2. Included from an example file

You can include source or configuration from an external file. In this way you can provide a working example configuration maintaining doc and example at the same time. The procedure and rules are the same as with images: the path is relative to the *.adoc file where the file to be used is included.

Include complete external file

```
[source,xml]
----
include::../configs/wmi-config.xml[]
----
```

This is how it's rendered:

```
<?xml version="1.0"?>
<wmi-config retry="2" timeout="1500"
  username="Administrator" domain="WORKGROUP" password="password">
</wmi-config>
```

6.10.3. Include parts of a file

To include a specific segment of a large configuration file, assign tags that tell AsciiDoc the section to include. In this example, just the service definition of the *ICMP monitor* should be included.

In the 'poller-configuration.xml', tag the section in the following manner:

```

...
<rrd step="300">
  <rra>RRA:AVERAGE:0.5:1:2016</rra>
  <rra>RRA:AVERAGE:0.5:12:1488</rra>
  <rra>RRA:AVERAGE:0.5:288:366</rra>
  <rra>RRA:MAX:0.5:288:366</rra>
  <rra>RRA:MIN:0.5:288:366</rra>
</rrd>
<!-- # tag::IcmpServiceConfig[] -->
<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>
<!-- # end::IcmpServiceConfig[] -->
<service name="DNS" interval="300000" user-defined="false" status="on">
  <parameter key="retry" value="2" />
  <parameter key="timeout" value="5000" />
  <parameter key="port" value="53" />
...

```

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>

```



Spaces and tabs are taken from the original file.

6.11. Cheat sheets and additional hints

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

- [AsciiDoc Writers Guide](#)
- [AciDoc Syntax Quick Reference](#)
- [AsciiDoc FAQ](#)
- [XPT AsciiDoc Cheatsheet](#)

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

- [AsciiDoc User Manual](#)
- [AsciiDoc Maven Plugin](#)
- [AsciiDoc discussion list](#)
- [AsciiDoc issue tracker](#)
- [Docbook to AsciiDoc](#)

6.12. Migrating content from project wiki

The [project wiki](#) contains much information that ought to be migrated to the official documentation set. To help with this effort, we have a wiki template which informs readers of articles that are tagged for migration to the official docs, or that have already been migrated. When you identify an article in the OpenNMS wiki whose information should be migrated (either in its entirety, or just individual sections), use the following process.

1. If you do not already have a wiki account, [request one](#) before getting started. Your request must be approved by a wiki admin. If you don't get approved within a day, send a note to the [opennms-devel mailing list](#) or on the [OpenNMS Development chat channel](#).
2. Create an issue in the [project issue tracker](#), in [project NMS](#). Note the issue number; you will use it below.
3. After logging in to the wiki, visit the article whose content should be migrated.
4. Click on the **Edit Source** link at the top of the article view.
5. Add text like the following to the top of the article source editor:

```
{{OfficialDocs | scope=article | guide=admin | issue=NMS-9926 | date=March 2018 |
completed=false}}
```

- The value of the `scope` attribute must be either `article`, if the entire article should be migrated, or `section` if only specific section(s) should be migrated.
 - When using `scope = section`, it's fine to use this template multiple times in a single article.
- The value of the `guide` attribute must be one of `admin`, `development`, `install`, or `user`.
 - If the information in an article should be migrated to multiple official guides, use multiple instances of the `{{OfficialDocs}}` template; try to target these by section when possible.
- The value of the `issue` parameter must be a valid issue ID in the [project issue tracker](#), and will become a live link

- The value of the `date` parameter should be the month and year when the tag was added, e.g. `March 2018`.
- The `completed` parameter is optional; it is assumed to be false if omitted, or true if its value is either `true` or `yes`.



Figure 3. Wiki source editor with example `OfficialDocs` template usage

6. Enter an edit summary such as **Tagged for migration to official docs, NMS-12345** and click **Show preview**
7. After verifying that your changes render as expected (see image), click **Save changes**.

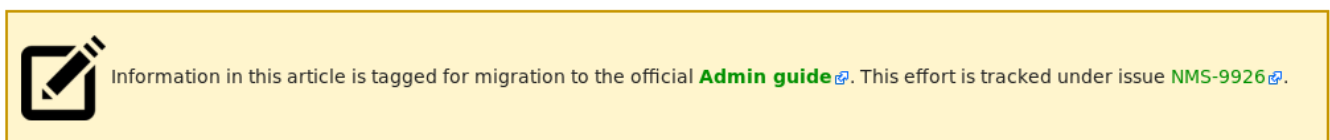


Figure 4. Rendering of `OfficialDocs` wiki template on an article pending migration

8. Migrate the information, making sure to follow the guidelines laid out earlier in this section; do not just copy and paste, and watch out for obsolete information. If you need help, contact the developers through one of the methods mentioned above.
9. Once the migration is complete and the issue is closed, edit the wiki article again and change `completed=false` to `completed=true`.
10. The rendering of the template will change to indicate that the migration has been completed.

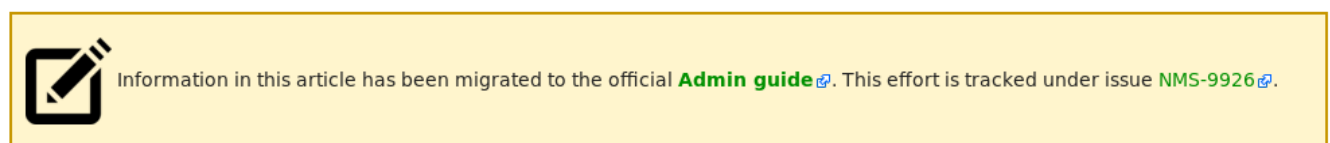


Figure 5. Rendering of `OfficialDocs` wiki template on an article whose migration is completed

Adding the `{{OfficialDocs}}` template to an article will implicitly add that article to a pair of wiki categories:

- **Migration to official docs pending** or **Migration to official docs completed**, according to the value of the `completed` attribute
- **Migrate to X guide**, according to the value of the `guide` attribute

Chapter 7. AMQP Integration

The AMQP Integration allows external systems to communicate with the event bus of OpenNMS Meridian and receive alarms via the AMQP protocol.



AMQP is standard messaging protocol supported by a number of brokers including *ActiveMQ* and *QPID*.

The integration is written using Camel + OSGi and has the following components:

- Event Forwarder
- Event Receiver
- Alarm Northbounder

Custom filtering (i.e. which events to forward) and transformations (i.e. how the events are represented in the messages) can be used in each of the components. Generic implementations



Each component can be configured and setup independently, i.e. you can choose to only forward alarms.

7.1. Event Forwarder

The event forwarder listens for *all* events on the internal event bus of OpenNMS Meridian. Events from the bus are sent to a Camel processor, which can filter or transform these, before being sent to the AMQP endpoint.

The event forwarder exposes the following properties via the `org.opennms.features.amqp.eventforwarder` pid:

Property	Default	Description
connectionUrl	amqp://localhost:5672	Used by the <code>JmsConnectionFactory</code> . See AMQP for details.
username	guest	Username
password	guest	Password
destination	amqp:topic:opennms-events	Target queue or topic. See AMQP for details.
processorName	default-event-forwarder-processor	Named <code>org.apache.camel.Processor</code> implementation used to filter and/or format the events.

The default processor, the `default-event-forwarder-processor`, marshalls events to XML and does

not perform any filtering. This means that when enabled, all events will be forwarded to the AMQP destination with XML strings as the message body.

7.1.1. Setup

Start by logging into a Karaf shell.

Update the properties with your deployment specific values:

```
config:edit org.opennms.features.amqp.eventforwarder
config:property-set connectionUrl amqp://localhost:5672
config:property-set destination amqp:topic:opennms-events
config:property-set processorName default-event-forwarder-processor
config:update
```

Install the feature:

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

7.1.2. Debugging

You can get detailed information on the Camel route using:

```
camel:route-info forwardEvent
```

7.2. Event Receiver

The event receiver listens for messages from an AMQP target and forwards them onto the internal event bus of OpenNMS Meridian. Messages are sent to a Camel processor, which can filter or transform these, before being sent onto the event bus.

The event receiver exposes the following properties via the `org.opennms.features.amqp.eventreceiver` pid:

Property	Default	Description
connectionUrl	amqp://localhost:5672	Used by the <code>JmsConnectionFactory</code> . See AMQP for details.
username	guest	Username
password	guest	Password
source	amqp:queue:opennms-events	Source queue or topic. See AMQP for details.

Property	Default	Description
processorName	default-event-receiver-processor	Named org.apache.camel.Processor implementation used to filter and/or format the events.

The default processor, the [default-event-receiver-processor](#), expects the message bodies to contain XML strings which are it unmarshalls to events.

7.2.1. Setup

Start by logging into a Karaf shell.

Update the properties with your deployment specific values:

```
config:edit org.opennms.features.amqp.eventreceiver
config:property-set connectionUrl amqp://localhost:5672
config:property-set source amqp:queue:opennms-events
config:property-set processorName default-event-receiver-processor
config:update
```

Install the feature:

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

7.2.2. Debugging

You can get detailed information on the Camel route using:

```
camel:route-info receiveEvent
```

7.3. Alarm Northbouncer

The alarm northbouncer listens for *all* northbound alarms. Alarms are sent to a Camel processor, which can filter or transform these, before being sent to the AMQP endpoint.

The alarm northbouncer exposes the following properties via the [org.opennms.features.amqp.alarmnorthbouncer](#) pid:

Property	Default	Description
connectionUrl	amqp://localhost:5672	Used by the JmsConnectionFactory . See AMQP for details.

Property	Default	Description
username	guest	Username
password	guest	Password
destination	amqp:topic:opennms-alarms	Target queue or topic. See AMQP for details.
processorName	default-alarm-northbunder-processor	Named <code>org.apache.camel.Processor</code> implementation used to filter and/or format the alarms.

The default processor, the `default-alarm-northbunder-processor`, converts the alarms to a string and does not perform any filtering. This means that when enabled, all alarms will be forwarded to the AMQP destination with strings as the message body.

7.3.1. Setup

Start by logging into a Karaf shell.

Update the properties with your deployment specific values:

```
config:edit org.opennms.features.amqp.alarmnorthbunder
config:property-set connectionUrl amqp://localhost:5672
config:property-set destination amqp:topic:opennms-alarms
config:property-set processorName default-alarm-northbunder-processor
config:update
```

Install the feature:

```
feature:install opennms-amqp-alarm-northbunder
```

7.3.2. Debugging

You can get detailed information on the Camel route using:

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

7.4. 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 in the Karaf container, you can update the loaded you can refer to your processor with:

```

config:edit org.opennms.features.amqp.eventforwarder
config:property-set 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

```

Chapter 8. Design and Styleguidelines

8.1. Jasper Report Guideline

Building and contributing JasperReports is a way to contribute to the project. To make it easier to maintain and style reports the following layout guideline can be used to have similar and more consistent report layout.

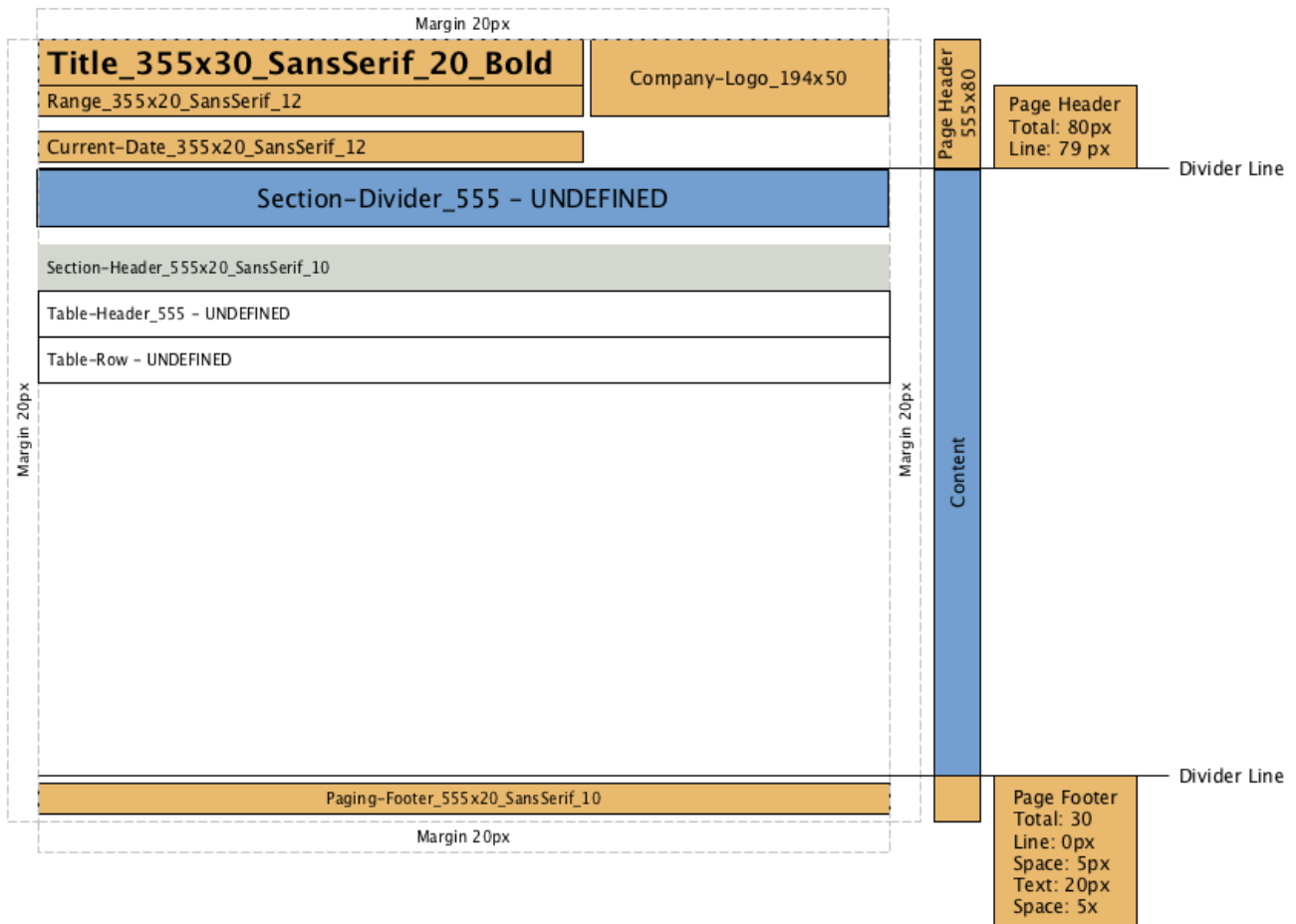


Figure 6. Layout for creating JasperReports

The following formatting can be applied:

Type	Convention
Date	yyyy/MM/dd HH:mm:ss
Report Range	Report Begin: \${startDate} Report End: \${endDate}
Paging	Page \${current} of \${total}



Based on this template definition there exist a [GitHub repository](#) which contains a JasperReport template.