Developers Guide

OpenNMS Meridian 2017.1.11, Last updated 2018-09-20 06:47:49 EDT

Table of Contents

Chapter 1. Setup a development system

This guide describes the requirements and the steps necessary in order to get started with the development of the OpenNMS project.

1.1. Operating System / Environment

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

- Linux Mint with Cinnamon Desktop environment
- Ubuntu Desktop
- Mac OS X

This documentation assumes that you chose a debian based desktop environment.

1.2. Installation

The next chapter describes the full setup of your environment in order to meet the pre-requirements. Simply follow these instructions, 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 OpenNMS successfully built, please follow the wiki Running OpenNMS.

1.3. Tooling

We recommend the following toolset:

- IDE: IntelliJ IDEA Ultimate
- DB-Tool: DBeaver or Postgres Admin - pgAdmin
- Graphing: yEd
- Other: atom.io

1.4. Useful links

1.4.1. General

- https://www.github.com/OpenNMS/opennms: The source code hosted on GitHub
- http://wiki.opennms.org: Our Wiki, especially the start page is of interest. It points you in the right directions.
- http://issues.opennms.org: Our issue/bug tracker.
- https://github.com/opennms-forge/vagrant-opennms-dev: A vagrant box to setup a virtual box to build OpenNMS
- https://github.com/opennms-forge/vagrant-opennms: A vagrant box to setup 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.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`.

Simple Java 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.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 + "].nodeSnmp[]" %}

{# 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">
             {{ ((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>
    <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: #}

            <img src="/opennms/graph/graph.png?resourceId=node[{{ node.id
}}].nodeSnmp[]&report=netsnmp.hrSystemUptime&start={{ start }}&end={{ end }}&
width=170&height=30"/>
        </td>
    </tr>
</table>
```

2.2. GraphML

In OpenNMS Meridian the `GraphMLTopoloyProvider` 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
`namespace`	yes	Graph	`string`	-	The namespace must be unique overall existing Topologies.
`description`	no	Graph	`string`	-	A description, which is shown in the Info Panel.
`preferred-layout`	no	Graph	`string`	`D3`	Defines a preferred layout.
`focus-strategy`	no	Graph	`string`	`FIRST`	Defines a focus strategy. See Focus Strategies for more information.
`focus-ids`	no	Graph	`string`	-	Refers to nodes ids in the graph. This is required if `focus-strategy` is `SPECIFIC`. If multiple ids should be add to focus, they are separated by `,`. Example: `node1,node2`
`semantic-zoom-level`	no	Graph	`int`	`1`	Defines the default SZL.
`vertex-status-provider`	no	Graph	`string`	-	Defines which Vertex Status Provider should be used, e.g. `default`, `script` or `propagate`
`iconKey`	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 id 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 foreignID. Please note that this attribute will not be used when the attribute nodeID 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 foreignSource. Please note that this attribute will not be used when the attribute nodeID is set.
tooltipText	no	Node, Edge	string		Defines a custom tooltip. If not defined, the id attribute is used instead.
level	no	Node	int	0	Sets the level of the Vertex which is used by certain layout algorithms i.e. Hierarchical Layout and Grid Layout.
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.

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>
```

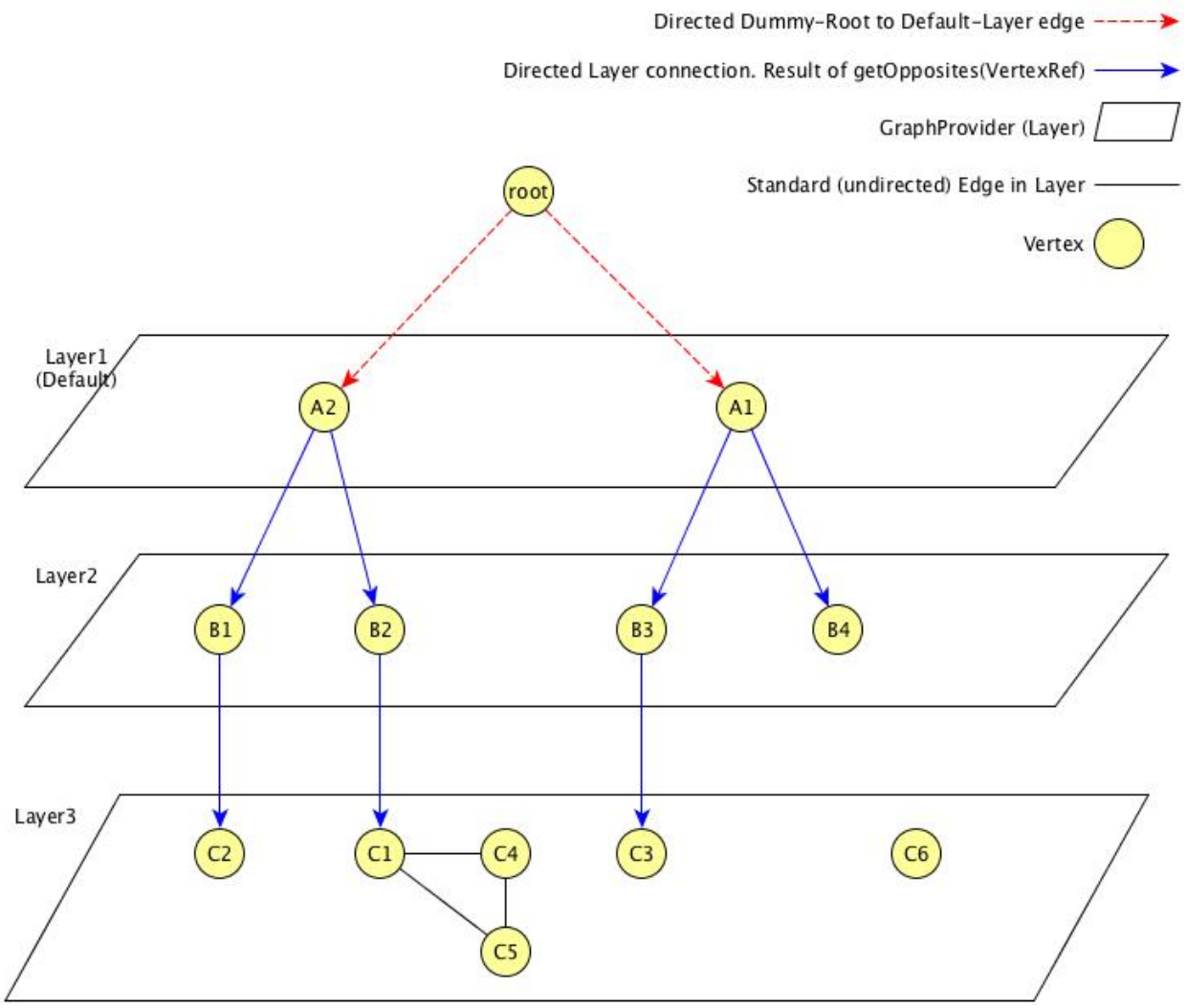
2.2.8. Breadcrumbs

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 algorithms 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`, than 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>
```

Chapter 3. CORS Support

3.1. 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

3.2. 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

3.3. 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 4. ReST API

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

4.1. ReST URL

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

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

4.2. 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.

4.3. 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`.

4.4. Standard Parameters

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

Table 2. ReST standard parameter for resources

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

Parameter	Description
`eq`	Checks for equality
`ne`	Checks for non-equality
`ilike`	Case-insensitive wildcarding (`%` is the wildcard)
`like`	Case-sensitive wildcarding (`%` is the wildcard)
`gt`	Greater than
`lt`	Less than
`ge`	Greater than or equal
`le`	Less than or equal

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

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

4.5. Standard filter examples

Take `/events` as an example.

Resource	Description
`/events?eventUei=uei.opennms.org/internal/rtc/subscribe`	would return the first 10 events with the rtc subscribe UEI, (10 being the default limit for events)
`/events?eventUei=uei.opennms.org/internal/rtc/subscribe&limit=0`	would return **all** the rtc subscribe events (potentially quite a few)
`/events?id=100&comparator=gt`	would return the first 10 events with an id greater than 100
`/events?eventAckTime=notnull`	would return the first 10 events that have a non-null Ack time (i.e. those that have been acknowledged)
`/events?eventAckTime=notnull&id=100&comparator=gt&limit=20`	would return the first 20 events that have a non-null Ack time and an id greater than 100. Note that the notnull value causes the comparator to be ignored for eventAckTime
`/events?eventAckTime=2008-07-28T04:41:30.530%2B12:00&id=100&comparator=gt&limit=20`	would return the first 20 events that have were acknowledged after 28th July 2008 at 4:41am (+12:00), and an id greater than 100. Note that the same comparator applies to both property comparisons. Also note that you must URL encode the plus sign when using GET.
`/events?orderBy=id&order=desc`	would return the 10 latest events inserted (probably, unless you've been messing with the id's)

4.6. 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).

4.7. 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
`node[ID]`	Identifies a node by its database ID, which is always an integer	`node[42]`
`node[FS:FID]`	Identifies a node by its foreign-source name and foreign-ID, joined by a single colon	`node[Servers:115da833-0957-4471-b496-a731928c27dd]`

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
`nodeSnmp[]`	Node-level (scalar) performance data for the node in question. This type is the only one where the instance identifier is empty.
`interfaceSnmp[eth0-04013f75f101]`	A layer-two interface as represented by a row in the SNMP `ifTable`. The instance identifier is composed of the interface's `ifName` and its `ifPhysAddress` (if it has one).
`dskIndex[_root_fs]`	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].nodeSnmp[]`
- `node[42].interfaceSnmp[eth0-04013f75f101]`
- `node[Servers:115da833-0957-4471-b496-a731928c27dd].dskIndex[_root_fs]`

4.8. Currently Implemented Interfaces

4.8.1. 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
`/acks`	Get a list of acknowledgements.
`/acks/count`	Get the number of acknowledgements. (Returns plaintext, rather than *XML* or *JSON*.)
`/acks/{id}`	Get the acknowledgement specified by the given *ID*.

POSTs (Setting Data)

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

Usage examples with curl

Acknowledge notification #3

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

Escalate alarm #42

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

4.8.2. Alarm Statistics

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

GETs (Reading Data)

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

4.8.3. 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.
`/alarms/count`	Get the number of alarms. (Returns plaintext, rather than *XML* or *JSON*.)
`/alarms/{id}`	Get the alarms specified by the given *ID*.

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

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

PUTs (Modifying Data)

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

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

New in OpenNMS 1.11.0

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

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

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

4.8.4. Events

GETs (Reading Data)

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

PUTs (Modifying Data)

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

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

Resource	Description
`/events?x=y&…&ack=''(true;false)`	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
`/events`	Publish an event on the event bus.

4.8.5. Categories

GETs (Reading Data)

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

POSTs (Adding Data)

Resource	Description
`/categories`	Adds a new category.

PUTs (Modifying Data)

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

DELETEs (Removing Data)

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

4.8.6. 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
`/foreignSources`	Get all active foreign sources.
`/foreignSources/default`	Get the active default foreign source.
`/foreignSources/deployed`	Get the list of all deployed (active) foreign sources.
`/foreignSources/deployed/count`	Get the number of deployed foreign sources. (Returns plaintext, rather than *XML* or *JSON*.)
`/foreignSources/{name}`	Get the active foreign source named {name}.
`/foreignSources/{name}/detectors`	Get the configured detectors for the foreign source named {name}.
`/foreignSources/{name}/detectors/{detector}`	Get the specified detector for the foreign source named {name}.
`/foreignSources/{name}/policies`	Get the configured policies for the foreign source named {name}.
`/foreignSources/{name}/policies/{policy}`	Get the specified policy for the foreign source named {name}.

POSTs (Adding Data)

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

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

PUTs (Modifying Data)

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

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

DELETEs (Removing Data)

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

4.8.7. Groups

Like users, groups have a simplified interface as well.

GETs (Reading Data)

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

POSTs (Adding Data)

Resource	Description
`/groups`	Add a new group.

PUTs (Modifying Data)

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

DELETEs (Removing Data)

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

4.8.8. Heatmap

GETs (Reading Data)

Resource	Description
`/heatmap/outages/categories`	Sizes and color codes based on outages for nodes grouped by *Surveillance Categories*
`/heatmap/outages/foreignSources`	Sizes and color codes based on outages for nodes grouped by *Foreign Source*
`/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 *Surveillance Category*
`/heatmap/outages/nodesByForeignSource/{foreignSource}`	Sizes and color codes based on outages for nodes associated with a specific *Foreign Source*
`/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 *Surveillance Categories*
`/heatmap/alarms/foreignSources`	Sizes and color codes based on alarms for nodes grouped by *Foreign Source*

Resource	Description
`/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 *Surveillance Category*
`/heatmap/alarms/nodesByForeignSource/{foreignSource}`	Sizes and color codes based on alarms for nodes associated with a specific *Foreign Source*
`/heatmap/alarms/nodesByMonitoredService/{monitoredService}`	Sizes and color codes based on alarms for nodes providing a specific monitored service

4.8.9. Categories

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
`/ifservices`	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
`/ifservices/`	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"
```

4.8.10. KSC Reports

GETs (Reading Data)

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

PUTs (Modifying Data)

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

POSTs (Creating Data)

Documentation incomplete see issue: NMS-7162

DELETEs (Removing Data)

Documentation incomplete see issue: NMS-7162

4.8.11. Maps

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

GETs (Reading Data)

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

POSTs (Adding Data)

Resource	Description
`/maps`	Add a map.

PUTs (Modifying Data)

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

DELETEs (Removing Data)

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

4.8.12. 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
`/measurements/{resourceId}/{attribute}`	Retrieve the measurements for a single attribute

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

name	default	comment
start	-14400000	Timestamp in milliseconds. If > 0, the 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.

name	default	comment
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>
  <timestamps>1425581100000</timestamps>
  <timestamps>1425581400000</timestamps>
  <timestamps>1425581700000</timestamps>
  ...
</query-response>
```

POSTs (Reading Data)

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

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

An example of the POST body is available bellow.

Usage examples with curl

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

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

Contents of report.json


```
{
    "start": 1425563626316,
    "end": 1425585226316,
    "step": 10000,
    "maxrows": 1600,
    "source": [
        {
            "aggregation": "AVERAGE",
            "attribute": "ifHCInOctets",
            "label": "ifHCInOctets",
            "resourceId": "nodeSource[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,
                ...
            ]
        }
    ]
}
```


4.8.13. Nodes

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

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

GETs (Reading Data)

Resource	Description
/nodes	Get a list of nodes. This includes the ID and node label.

Resource	Description
/nodes/{id}	Get a specific node, by ID.
/nodes/{id}/ipinterfaces	Get the list of IP interfaces associated with the given node.
/nodes/{id}/ipinterfaces/{ipAddress}	Get the IP interface for the given node and IP address.
/nodes/{id}/ipinterfaces/{ipAddress}/services	Get the list of services associated with the given node and IP interface.
/nodes/{id}/ipinterfaces/{ipAddress}/services/{service}	Get the requested service associated with the given node, IP interface, and service name.
/nodes/{id}/snmpinterfaces	Get the list of SNMP interfaces associated with the given node.
/nodes/{id}/snmpinterfaces/{ifIndex}	Get the specific interface associated with the given node and ifIndex.
/nodes/{id}/categories	Get the list of categories associated with the given node.
/nodes/{id}/categories/{categoryName}	Get the category associated with the given node and category name.
/nodes/{id}/assetRecord	Get the asset record associated with the given node.

POSTs (Adding Data)

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

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

PUTs (Modifying Data)

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

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

Resource	Description
`/nodes/{id}/categories/{categoryName}`	Modify the category with the given node ID and name.

DELETEs (Removing Data)

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

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.

4.8.14. Notifications

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

GETs (Reading Data)

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

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

4.8.15. Outage Timelines

GETs (Reading Data)

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

4.8.16. Outages

GETs (Reading Data)

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

4.8.17. 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
`/requisitions`	Get all active requisitions.
`/requisitions/count`	Get the number of active requisitions. (Returns plaintext, rather than *XML* or *JSON*.)
`/requisitions/deployed`	Get the list of all deployed (active) requisitions.
`/requisitions/deployed/count`	Get the number of deployed requisitions. (Returns plaintext, rather than *XML* or *JSON*.)
`/requisitions/{name}`	Get the active requisition for the given foreign source name.
`/requisitions/{name}/nodes`	Get the list of nodes being requisitioned for the given foreign source name.
`/requisitions/{name}/nodes/{foreignId}`	Get the node with the given foreign *ID* for the given foreign source name.

Resource	Description
`/requisitions/{name}/nodes/{foreignId}/interfaces`	Get the interfaces for the node with the given foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}`	Get the interface with the given IP for the node with the specified foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}/services`	Get the services for the interface with the specified IP address, foreign *ID*, and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/interfaces/{ipAddress}/services/{service}`	Get the given service with the specified IP address, foreign *ID*, and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/categories`	Get the categories for the node with the given foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/categories/{categoryName}`	Get the category with the given name for the node with the specified foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/assets`	Get the assets for the node with the given foreign *ID* and foreign source name.
`/requisitions/{name}/nodes/{foreignId}/assets/{assetName}`	Get the value of the asset for the given assetName for the node with the given foreign *ID* and foreign source name.

POSTs (Adding Data or Updating existing Data)

Expects JSON/XML

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

PUTs (Modifying Data)

Expects form-urlencoded

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

DELETEs (Removing Data)

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

4.8.18. 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
`/resources`	Retrieve the full tree of resources in the system (expensive, use with care)

Resource	Description
`/resources/{resourceId}`	Retrieve the tree of resources starting with the named resource ID
`/resources/fornode/{nodeCriteria}`	Retrieve the tree of resources for a node, given its database ID or `foreign-source:foreign-ID` tuple

DELETEs (Removing Data)

Resource	Description
`/resources/{resourceId}`	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>
      <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&amp;ifindex=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"
```

4.8.19. 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

Resource	Description
/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
```

4.8.20. Scheduled Outages

GETs (Reading Data)

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

POSTs (Setting Data)

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

PUTs (Modifying Data)

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

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

DELETEs (Removing Data)

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

4.8.21. SNMP Configuration

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

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

GETs (Reading Data)

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

PUTs (Modifying Data)

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

Determine API version

To determine the version of the *API* running in your OpenNMS 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 "&lt;snmp-info&gt;
             &lt;community&gt;yRuSonoZ&lt;/community&gt;
             &lt;port&gt;161&lt;/port&gt;
             &lt;retries&gt;1&lt;/retries&gt;
             &lt;timeout&gt;2000&lt;/timeout&gt;
             &lt;version&gt;v2c&lt;/version&gt;
         &lt;/snmp-info&gt;" \
     -u admin:admin http://localhost:8980/opennms/rest/snmpConfig/10.1.1.1
```

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

Example 2:

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

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

API Version 2

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

The interface uses *SnmpInfo* objects for communication. Therefore it is possible to set for example v1 and v3 parameters in one request (e.g. `readCommunity` String and `privProtocol` String). However OpenNMS 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 *SnmpInfo* schema is defined as follows:

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

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

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

attribute snmp-info.xml	**attribute snmp-config.xml**
default	restricted to version
restriction	version

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
read-community	public

attribute snmp-info.xml	attribute snmp-config.xml
v1, v2c	
writeCommunity	write-community
private	v1, v2c
	securityName
security-name	opennmsUser
v3	
securityLevel	security-level
noAuthNoPriv	v3

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

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

attribute snmp-info.xml	attribute snmp-config.xml
	contextName
context-name	
v3	
enterpriseId	enterprise-id
	v3

Example 1:

```
curl -v -X PUT -H "Content-Type: application/xml" \
     -H "Accept: application/xml" \
     -d "&lt;snmp-info&gt;
             &lt;readCommunity&gt;yRuSonoZ&lt;/readCommunity&gt;
             &lt;port&gt;161&lt;/port&gt;
             &lt;retries&gt;1&lt;/retries&gt;
             &lt;timeout&gt;2000&lt;/timeout&gt;
             &lt;version&gt;v2c&lt;/version&gt;
         &lt;/snmp-info&gt;" \
     -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 "&lt;snmp-info&gt;
             &lt;readCommunity&gt;yRuSonoZ&lt;/readCommunity&gt;
             &lt;port&gt;161&lt;/port&gt;
             &lt;retries&gt;1&lt;/retries&gt;
             &lt;timeout&gt;2000&lt;/timeout&gt;
             &lt;version&gt;v1&lt;/version&gt;
             &lt;securityName&gt;secret-stuff&lt;/securityName&gt;
             &lt;engineId&gt;engineId&lt;/engineId&gt;
         &lt;/snmp-info&gt;" \
     -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 "&lt;snmp-info&gt;
             &lt;readCommunity&gt;yRuSonoZ&lt;/readCommunity&gt;
             &lt;port&gt;161&lt;/port&gt;
             &lt;retries&gt;1&lt;/retries&gt;
             &lt;timeout&gt;2000&lt;/timeout&gt;
             &lt;version&gt;v3&lt;/version&gt;
             &lt;securityName&gt;secret-stuff&lt;/securityName&gt;
             &lt;engineId&gt;engineId&lt;/engineId&gt;
         &lt;/snmp-info&gt;" \
     -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`.

4.8.22. 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)

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

POSTs (Adding Data)

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

PUTs (Modifying Data)

Parameter	Description
`/users/{username}`	Update an existing user's full-name, user-comments, password, passwordSalt and duty-schedule values.
`/users/{username}/roles/{rolename}`	Add a security role to the user. (new in OpenNMS 19)

DELETEs (Removing Data)

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

4.8.23. SNMP Trap Northbounder Interface Configuration

GETs (Reading Data)

Resource	Description
`/config/snmptrap-nbi`	Gets full content of the configuration.
`/config/snmptrap-nbi/status`	Gets the status of the SNMP Trap NBI (returns either true or false).

Resource	Description
`/config/snmptrap-nbi/destinations`	Gets the name of all the existing destinations.
`/config/snmptrap-nbi/destinations/{name}`	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
`/config/snmptrap-nbi/status?enabled=(true;false)`	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
`/config/snmptrap-nbi`	Updates the full content of the configuration.
`/config/snmptrap-nbi/destinations`	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
`/config/snmptrap-nbi/destinations/{name}`	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
`/config/snmptrap-nbi/destinations/{name}`	Updates the content of the destination named {name}

4.8.24. Email Northbounder Interface Configuration

GETs (Reading Data)

Resource	Description
`/config/email-nbi`	Gets full content of the configuration.
`/config/email-nbi/status`	Gets the status of the Email NBI (returns either true or false).
`/config/email-nbi/destinations`	Gets the name of all the existing destinations.
`/config/email-nbi/destinations/{name}`	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
`/config/email-nbi/status?enabled=(true;false)`	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
`/config/email-nbi/destinations`	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
`/config/email-nbi/destinations/{name}`	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
`/config/email-nbi/destinations/{name}`	Updates the content of the destination named {name}

4.8.25. Javamail Configuration

GETs (Reading Data)

Resource	Description
`/config/javamail/default/readmail`	Get the name of the default readmail config.
`/config/javamail/default/sendmail`	Get the name of the default sendmail config.
`/config/javamail/readmails`	Get the name of all the existing readmail configurations.
`/config/javamail/sendmails`	Get the name of all the existing sendmail configurations.
`/config/javamail/end2ends`	Get the name of all the existing end2end mail configurations.
`/config/javamail/readmails/{name}`	Get the content of the readmail configuration named {name}
`/config/javamail/sendmails/{name}`	Get the content of the sendmail configuration named {name}
`/config/javamail/end2ends/{name}`	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
`/config/javamail/readmails`	Adds a new or overrides an existing readmail configuration.
`/config/javamail/sendmails`	Adds a new or overrides an existing sendmail configuration.
`/config/javamail/end2ends`	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
`config/javamail/default/readmail/{name}`	Sets the readmail named {name} as the new default.
`config/javamail/default/sendmail/{name}`	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
/config/javamail/readmails/{name}	Updates the content of the readmail configuration named {name}
/config/javamail/sendmails/{name}	Updates the content of the sendmail configuration named {name}
/config/javamail/end2ends/{name}	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
/config/javamail/readmails/{name}	Removes the readmail configuration named {name}
/config/javamail/sendmails/{name}	Removes the sendmail configuration named {name}
/config/javamail/end2ends/{name}	Removes the end2end mail configuration named {name}

4.8.26. Syslog Northbounder Interface Configuration

GETs (Reading Data)

Resource	Description
/config/syslog-nbi	Gets full content of the configuration.
/config/syslog-nbi/status	Gets the status of the Syslog NBI (returns either true or false).
/config/syslog-nbi/destinations	Gets the name of all the existing destinations.
/config/syslog-nbi/destinations/{name}	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
/config/syslog-nbi/status?enabled=(true;false)	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
`/config/syslog-nbi`	Updates the full content of the configuration.
`/config/syslog-nbi/destinations`	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
`/config/syslog-nbi/destinations/{name}`	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
`/config/syslog-nbi/destinations/{name}`	Updates the content of the destination named {name}

4.8.27. 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>
      <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
`/opennms/api/v2/business-services`	Provides a brief list of all defined *Business Services*
`/opennms/api/v2/business-services/{id}`	Returns the *Business Service* identified by `id` included the current operational state
`/opennms/api/v2/business-services/egdes/{edgeId}`	Returns the edge of the *Business Service* identified by `edgeId`

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

PUTs (Modifying Data)

Resource	Description
`/opennms/api/v2/business-services/{id}`	Modifies an existing *Business Service* identified by `id`

POSTs (Adding Data)

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

DELETEs (Removing Data)

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

4.8.28. 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
```

4.9. ReST API Examples

4.9.1. 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";
     }
}
```

4.9.2. 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.

4.9.3. 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
```

4.9.4. 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 5. 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.

5.1. File Structure in opennms-doc

Directory	Contents
`guide-user/`	module with the guide for OpenNMS user e.g. NOC user who don't change behavior of OpenNMS.
`guide-admin/`	module with the guide for administrators configuring, optimizing and running OpenNMS
`guide-development/`	module with the guide for those who want to develop OpenNMS
`guide-install/`	module with the guide of how to install OpenNMS on different operating systems
`releasenotes/`	module with the changelog and release notes

5.2. 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: issue tracker for documentation!

5.2.1. 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 indicate 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'.

5.2.2. 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 being issued!;
- Forcing line breaks can be achieved 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

5.3. Headings and document structure

Each document starts over with headings from level zero (the document title). Each document should have an id. In some cases sections in the document need to have id's as well, 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, the leveloffset attribute is used.

Subsequent headings in a document should use the following syntax:

```
== Subheading

... content here ...

=== Subsubheading

content here ...
```

5.4. Links

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

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

This will render as: [doc-guidelines-links]

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

see <<target-id>>

this should suffice for most cases.

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

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

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

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

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

This will render in the output as: Link text here

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

```
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.

5.5. Admonitions and useful notes

These are useful for defining specific sections, such as Notes, Tips and Important information. We encourage the use of them 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:

This is my tip.

Source for giving a important hint

```
IMPORTANT: This is my important hint.
```

This is how its rendered:

This is my important hint.

Source for giving a caution

```
CAUTION: This is my caution.
```

This is how its rendered:

This is my caution.

Source for giving a warning

```
WARNING: This is my warning.
```

This is how its rendered:

This is my warning.

A multiline variation:

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

Which is rendered as:

Tiptext.
Line 2.

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

5.6. Attributes

Common attributes you can use in documents:

- {opennms-version} - rendered as "2017.1.11"

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

Sample Asciidoc attributes which can be used:

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

5.7. Comments

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

To write a comment:

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

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

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

5.8. Tables

For representing structured information you can use tables. 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 in order to give 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	`-`

5.9. Include images

When visualizing complex problems you can help the explanation and provide greater information by using an image. We use in OpenNMS documentation modules two directories for images.

The image folder structure mirrors the text structure. In this case it is a little bit easier to locate the AsciiDoc text file where the image is included.

Example folder structure for image files

```
.
└── opennms-doc①
    └── guide-doc②
        ├── README.adoc
        ├── pom.xml
        ├── src③
        │   └── asciidoc④
        │       ├── configs
        │       │   └── poller-configuration.xml
        │       ├── images⑤
        │       │   ├── 01_opennms-logo.png⑥
        │       │   └── 02_pris-overview.png
        │       ├── images_src⑦
        │       │   └── pris-overview.graphml⑧
        │       ├── index.adoc⑨
        │       └── text
        │           ├── images.adoc⑩
        │           ├── include-source.adoc
        │           ├── introduction.adoc
        │           └── writing.adoc
        └── target⑪
```

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 within. 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.

5.10. 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.

5.10.1. Explicitly defined in the document

be careful to use this kind of code snippets 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/.

5.10.2. 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>
```

5.10.3. Include parts of a file

If you want to include just a specific segment of a large configuration file, you can assign tags that indicate to AsciiDoc the section that is to be included. 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.

5.11. Cheat Sheets and additional hints

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

- readme

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

- Aciidoc Reference
- AsciiDoc FAQ
- AsciiDoc cheatsheet
- AsciiDoc Cheatsheet

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

- AsciiDoc User Manual
- AsciiDoc Maven Plugin
- AsciiDoc discussion list
- AsciiDoc issue tracker
- Docbook to AsciiDoc
- How to create handsome PDF documents without frustration

5.12. 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.

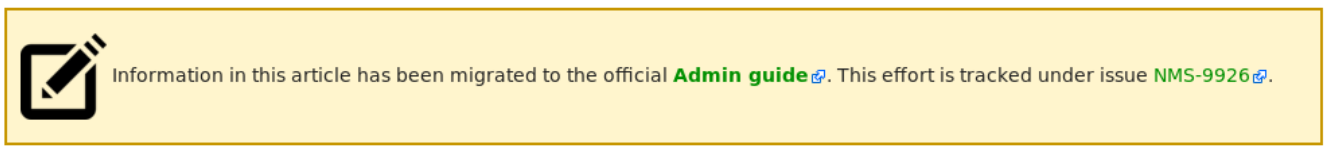


Figure 5. Rendering of `OfficialDocs` wiki template on an article whose migration is completed

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 6. 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

The integration is written using Camel + OSGi and exposes interfaces through which events and alarms can be filtered and/or transformed. The features are described in detail bellow.

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

6.1. 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 forwarer exposes the following properties via the `org.opennms.features.amqp.eventforwarder` pid:

Property	Default	Description
connectionUrl	amqp://guest:guest@onms/test?brokerlist=\'tcp://127.0.0.1:5672'	Used by the AMQPConnectionFactory. See ConnectionURL for a full list of options.
destination	amqp:OpenNMS-Exchange/opennms-routing-key	Target queue or topic. See AMQP for details.
processorName	default-event-forwarder-processor	Name `org.apache.camel.Processor` 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.

6.1.1. Setup

Start by logging into a Karaf shell.

Update the properties with your deployment specific values:

```
config:edit org.opennms.features.amqp.eventforwarder
propset connectionUrl amqp://guest:guest@onms/test?brokerlist=\'tcp://127.0.0.1:5672\'
propset destination amqp:OpenNMS-Exchange/opennms-routing-key
propset processorName default-event-forwarder-processor
config:update
```

Install the feature:

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

6.1.2. Debugging

You can get detailed information on the Camel route using:

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

6.2. 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 forwarer exposes the following properties via the `org.opennms.features.amqp.eventreceiver` pid:

Property	Default	Description
connectionUrl	amqp://guest:guest@onms/test?brokerlist=\'tcp://127.0.0.1:5672'	Used by the AMQPConnectionFactory. See ConnectionURL for a full list of options.
source	amqp:OpenNMS-Queue	Source queue or topic. See AMQP for details.
processorName	default-event-receiver-processor	Name `org.apache.camel.Processor` 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.

6.2.1. Setup

Start by logging into a Karaf shell.

Update the properties with your deployment specific values:

```
config:edit org.opennms.features.amqp.eventreceiver
propset connectionUrl amqp://guest:guest@onms/test?brokerlist=\'tcp://127.0.0.1:5672\'
propset source amqp:OpenNMS-Queue
propset processorName default-event-receiver-processor
config:update
```

Install the feature:

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

6.2.2. Debugging

You can get detailed information on the Camel route using:

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

6.3. Alarm Northbounder

The alarm northbounder 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 northbounder exposes the following properties via the `org.opennms.features.amqp.alarmnorthbounder` pid:

Property	Default	Description
connectionUrl	`amqp://guest:guest@onms/test?brokerlist=\'tcp://127.0.0.1:5672'`	Used by the AMQPConnectionFactory. See ConnectionURL for a full list of options.
destination	`amqp:OpenNMS-Exchange/opennms-routing-key`	Target queue or topic. See AMQP for details.
processorName	`default-alarm-northbounder-processor`	Name `org.apache.camel.Processor` used to filter and/or format the events.

The default processor, the `default-alarm-northbounder-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.

6.3.1. Setup

Start by logging into a Karaf shell.

Update the properties with your deployment specific values:

```
config:edit org.opennms.features.amqp.alarmnorthbounder
propset connectionUrl amqp://guest:guest@onms/test?brokerlist=\'tcp://127.0.0.1:5672\'
propset destination amqp:OpenNMS-Exchange/opennms-routing-key
propset processorName default-alarm-northbounder-processor
config:update
```

Install the feature:

```
features:install opennms-amqp-alarm-northbounder
```

6.3.2. Debugging

You can get detailed information on the Camel route using:

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

6.4. 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 registiry 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
propset processorName evilcorp-event-forwarder-processor
config:update
```

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

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

Chapter 7. Design and Styleguidelines

7.1. 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.

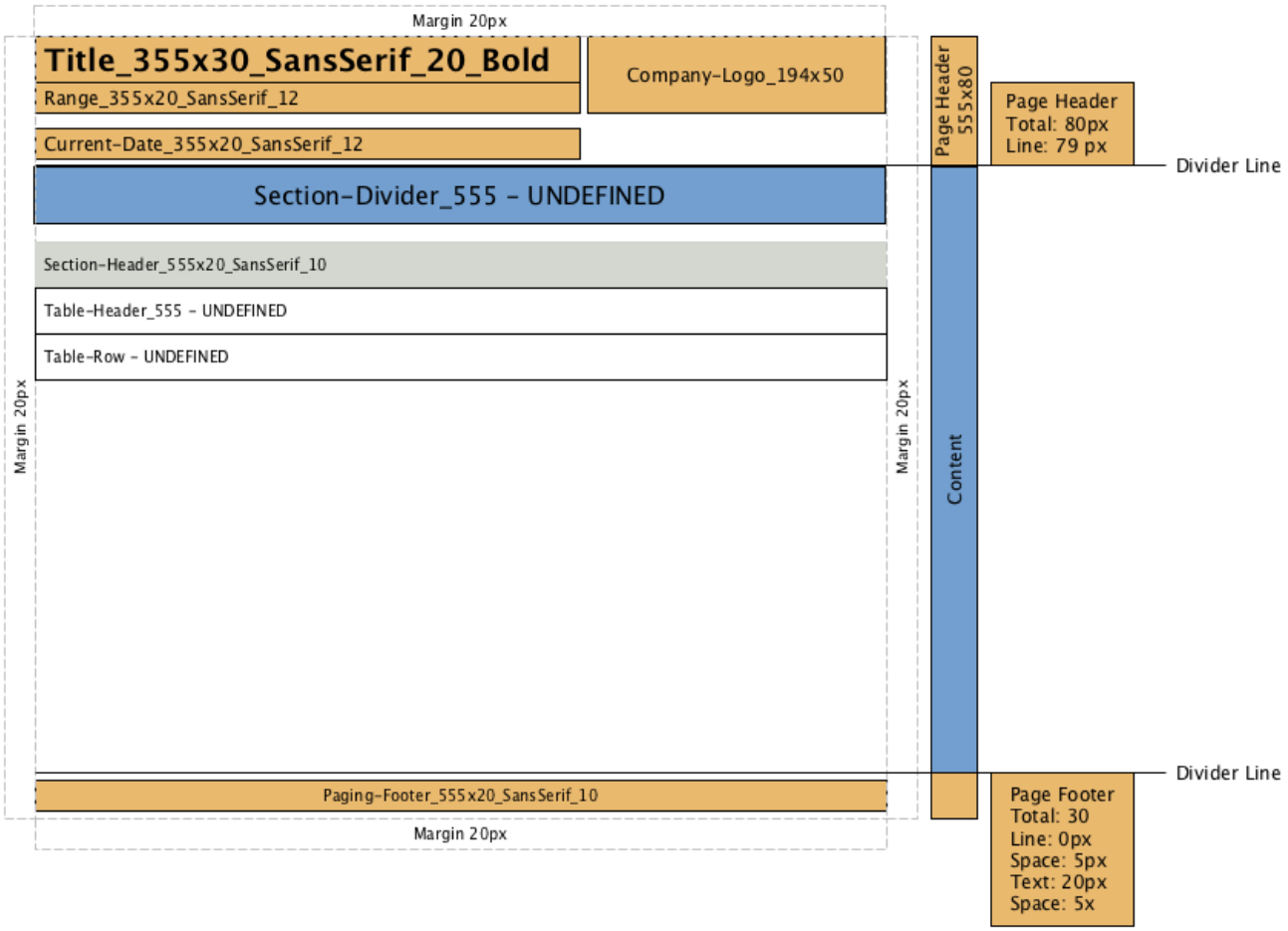


Figure 6. Layout for creating JasperReports

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.