Administrators Guide

Copyright (c) 2015-2018 The OpenNMS Group, Inc.

OpenNMS Meridian 2017.1.6, Last updated 2018-03-15 14:26:05 EDT

Table of Contents

| 1. | Data Choices | 1 |
|----|---------------------------------------|------|
| 2. | User Management | 2 |
| | 2.1. Users | 2 |
| | 2.2. Security Roles | 3 |
| | 2.3. Web UI Pre-Authentication | 6 |
| | 2.3.1. Enabling Pre-Authentication | 6 |
| | 2.3.2. Configuring Pre-Authentication | 7 |
| 3. | Administrative Webinterface | 8 |
| | 3.1. Grafana Dashboard Box | 8 |
| | 3.2. Operator Board | . 10 |
| | 3.2.1. Configuration | . 11 |
| | 3.2.2. Dashlets | . 13 |
| | 3.2.3. Boosting <i>Dashlet</i> | . 17 |
| | 3.2.4. Criteria Builder | . 17 |
| | 3.3. JMX Configuration Generator | . 18 |
| | 3.3.1. Web based utility | . 18 |
| | 3.3.2. CLI based utility | . 22 |
| | 3.4. Heatmap | . 27 |
| | 3.5. Trend | . 31 |
| 4. | Service Assurance | . 35 |
| | 4.1. Pollerd Configuration | . 35 |
| | 4.2. Critical Service | . 37 |
| | 4.3. Downtime Model | . 37 |
| | 4.4. Path Outages | . 38 |
| | 4.5. Poller Packages | . 39 |
| | 4.5.1. Response Time Configuration | . 40 |
| | 4.5.2. Overlapping Services | . 41 |
| | 4.5.3. Test Services on manually | . 43 |
| | 4.6. Service monitors | . 45 |
| | 4.6.1. AvailabilityMonitor | . 45 |
| | 4.6.2. BgpSessionMonitor | . 46 |
| | 4.6.3. BSFMonitor | . 49 |
| | 4.6.4. CiscoIpSlaMonitor | . 56 |
| | 4.6.5. CiscoPingMibMonitor | . 58 |
| | 4.6.6. CitrixMonitor | . 64 |
| | 4.6.7. DhcpMonitor | . 65 |
| | 4.6.8. DiskUsageMonitor | . 69 |
| | 4.6.9. DnsMonitor | . 71 |

| | 4.6.10. DNSResolutionMonitor | . 73 |
|-------|-------------------------------------|------|
| | 4.6.11. FtpMonitor | . 76 |
| | 4.6.12. HostResourceSwRunMonitor | . 77 |
| | 4.6.13. HttpMonitor | . 79 |
| | 4.6.14. HttpPostMonitor | . 83 |
| | 4.6.15. HttpsMonitor | . 85 |
| | 4.6.16. IcmpMonitor | . 85 |
| | 4.6.17. ImapMonitor | . 87 |
| | 4.6.18. ImapsMonitor | . 88 |
| | 4.6.19. JCifsMonitor | . 89 |
| | 4.6.20. JDBCMonitor | . 91 |
| | 4.6.21. JDBCStoredProcedureMonitor | . 92 |
| | 4.6.22. JDBCQueryMonitor | . 94 |
| | 4.6.23. JmxMonitor | . 97 |
| | 4.6.24. JolokiaBeanMonitor | . 98 |
| | 4.6.25. LdapMonitor | 100 |
| | 4.6.26. LdapsMonitor | 101 |
| | 4.6.27. MemcachedMonitor | 102 |
| | 4.6.28. NetScalerGroupHealthMonitor | 103 |
| | 4.6.29. NrpeMonitor | 105 |
| | 4.6.30. NtpMonitor | 106 |
| | 4.6.31. OmsaStorageMonitor | 107 |
| | 4.6.32. OpenManageChassisMonitor | 109 |
| | 4.6.33. PageSequenceMonitor | 110 |
| | 4.6.34. PercMonitor | 119 |
| | 4.6.35. Pop3Monitor | 120 |
| | 4.6.36. PrTableMonitor | 121 |
| | 4.6.37. RadiusAuthMonitor | 122 |
| | 4.6.38. SmbMonitor | 124 |
| | 4.6.39. SnmpMonitor | 125 |
| | 4.6.40. SshMonitor | 133 |
| | 4.6.41. SSLCertMonitor | 134 |
| | 4.6.42. StrafePingMonitor | 136 |
| | 4.6.43. TcpMonitor | 139 |
| | 4.6.44. SystemExecuteMonitor | 140 |
| | 4.6.45. VmwareCimMonitor | 141 |
| | 4.6.46. VmwareMonitor | 143 |
| | 4.6.47. Win32ServiceMonitor | 144 |
| | 4.6.48. WsManMonitor | 146 |
| | 4.6.49. XmpMonitor | 147 |
| 5. Pe | erformance Management | 149 |

| | 5.1. Collectd Configuration | 149 |
|----|--|-----|
| | 5.2. Collection Packages | 150 |
| | 5.2.1. Service Configurations | 151 |
| | 5.3. Collectors | 152 |
| | 5.3.1. JmxCollector | 152 |
| | 5.3.2. SnmpCollector | 154 |
| | 5.3.3. WS-Management | 156 |
| | 5.4. Stress Testing | 163 |
| | 5.5. Stress Testing | 163 |
| | 5.5.1. Interpreting the output | 163 |
| 6. | Events | 165 |
| | 6.1. Anatomy of an Event. | 165 |
| | 6.2. Sources of Events | 165 |
| | 6.2.1. SNMP Traps | 166 |
| | 6.2.2. Syslog Messages | 166 |
| | 6.2.3. TL1 Autonomous Messages | 168 |
| | 6.2.4. XML-TCP | 168 |
| | 6.2.5. ReST | 168 |
| | 6.3. The Event Bus | 168 |
| | 6.3.1. Associate an Event to a given node | 168 |
| 7. | Alarms | 169 |
| | 7.1. Alarm Sounds | 169 |
| | 7.2. Flashing Unacknowledged Alarms | 170 |
| | 7.3. Configuring Alarm Sounds and Flashing | 170 |
| 8. | Notifications | 172 |
| | 8.1. Introduction | 172 |
| | 8.2. Getting Started | 172 |
| | 8.2.1. Enabling Notifications | 172 |
| | 8.2.2. Configuring Destination Paths | 172 |
| | 8.2.3. Configuring Event Notifications | 173 |
| | 8.3. Concepts | 173 |
| | 8.3.1. Events and UEIs | 173 |
| | 8.3.2. Users, Groups, and On-Call Roles | 173 |
| | 8.3.3. Duty Schedules | 175 |
| | 8.3.4. Destination Paths | 175 |
| | 8.3.5. Notification Commands | 176 |
| | 8.4. Bonus Notification Methods | 177 |
| | 8.4.1. Mattermost | |
| | 8.4.2. Slack Notifications | |
| 9. | Provisioning | 180 |
| | 9.1. Introduction | |
| | | |

| | 9.2. Concepts | 180 |
|----|--|-----|
| | 9.2.1. Terminology | 181 |
| | 9.2.2. Addressing Scalability | 182 |
| | 9.3. Getting Started | 185 |
| | 9.3.1. Provisioning the SNMP Configuration | 185 |
| | 9.3.2. Automatic Discovery | 191 |
| | 9.3.3. Enhanced Directed Discovery | 191 |
| | 9.4. Import Handlers | 193 |
| | 9.4.1. File Handler | 193 |
| | 9.4.2. HTTP Handler | 193 |
| | 9.4.3. DNS Handler | 193 |
| | 9.5. Provisioning Examples | 196 |
| | 9.5.1. Basic Provisioning | 196 |
| | 9.5.2. Advanced Provisioning Example | 203 |
| | 9.6. Adapters | 218 |
| | 9.6.1. DDNS Adapter | 218 |
| | 9.6.2. RANCID Adapter | 218 |
| | 9.7. Integrating with Provisiond | 218 |
| | 9.7.1. Provisioning Groups of Nodes | 218 |
| | 9.7.2. Example | 218 |
| | 9.8. Provisioning Single Nodes (Quick Add Node) | 221 |
| | 9.9. Fine Grained Provisioning Using <i>provision.pl</i> | 221 |
| | 9.9.1. Create a new requisition | 221 |
| | 9.10. Yet Other API Examples | 223 |
| | 9.11. Service Detectors | 223 |
| | 9.11.1. SNMP Detector | 223 |
| 10 | . Business Service Monitoring | 226 |
| | 10.1. Business Service Definition | 226 |
| | 10.2. Edges | 227 |
| | 10.2.1. Child Services | 228 |
| | 10.2.2. IP Services | 228 |
| | 10.2.3. Custom Reduction Key | 228 |
| | 10.3. Map Functions | 229 |
| | 10.4. Reduce Functions | 229 |
| | 10.5. Business Service Daemon | 230 |
| 11 | . Topology Map | 232 |
| | 11.1. Properties | 232 |
| | 11.2. Icons | 232 |
| | 11.2.1. Icon resolution | 233 |
| | 11.2.2. Change existing icon mappings | 234 |
| | 11.2.3. Add new icons | 234 |

| 12. Asset Topology Provider | 37 |
|--|----|
| 12.1. Overview | 37 |
| 12.2. Asset layers | 41 |
| 12.3. Node filtering | 41 |
| 12.4. Configuration | 42 |
| 12.5. Creating Asset Based Topologies From Karaf Consol | 44 |
| 12.6. Creating Asset Based Topologies Using <i>OpenNMS Meridian</i> events | 45 |
| 12.7. Viewing the topology | 46 |
| 12.8. additional notes | 46 |
| 13. Database Reports | 47 |
| 13.1. Overview | 47 |
| 13.2. Modify existing reports | 47 |
| 13.3. Add a custom report | 48 |
| 13.4. Usage of Jaspersoft Studio | 48 |
| 13.4.1. Connect to the OpenNMS Meridian Database | 49 |
| 13.4.2. Use Measurements Datasource and Helpers | 49 |
| 13.5. Accessing Performance Data | 50 |
| 13.5.1. Fields | 51 |
| 13.5.2. Parameters | 51 |
| 13.6. Helper methods | 52 |
| 13.6.1. Usage of the interface descriptor | 53 |
| 13.6.2. Usage of the node source descriptor | 54 |
| 13.6.3. Usage of the interface descriptor | 55 |
| 13.6.4. Use HTTPS | 56 |
| 13.7. Limitations | 56 |
| 14. Enhanced Linkd | 57 |
| 14.1. Enlinkd Daemon | 57 |
| 14.2. Layer 2 Link Discovery | 58 |
| 14.2.1. LLDP Discovery | 59 |
| 14.2.2. CDP Discovery | 62 |
| 14.2.3. Transparent Bridge Discovery | |
| 14.3. Layer 3 Link Discovery | |
| 14.3.1. OSPF Discovery | 71 |
| 14.3.2. IS-IS Discovery | 72 |
| 15. Operation | 75 |
| 15.1. HTTPS / SSL | |
| 15.1.1. Standalone HTTPS with Jetty | 75 |
| 15.1.2. OpenNMS Meridian as HTTPS client | |
| 15.1.3. Differences between Java Trust Store and Java Key Store | |
| 15.1.4. Debugging / Properties | |
| 15.2. Geocoder Service | |

| 15.3. resourcecli: simple resource management tool | 278 |
|---|-----|
| 15.3.1. Usage | 278 |
| 15.3.2. Sub-command: list | 279 |
| 15.3.3. Sub-command: show | 280 |
| 15.3.4. Sub-command: delete | 280 |
| 15.4. newts-repository-converter: Rrd/Jrb to Newts migration utility | 281 |
| 15.4.1. Migration | 281 |
| 15.4.2. Usage | 282 |
| 15.4.3. Example 1: convert Rrd-based data with storeByGroup enabled | 282 |
| 15.4.4. Example 2: convert JRobin-based data with storeByGroup disabled | 283 |
| 15.5. Newts | 283 |
| 15.5.1. Configuration | 283 |
| 15.5.2. Cassandra Monitoring | 285 |
| 15.5.3. Newts Monitoring | 290 |
| 15.6. Daemon Configuration Files | 292 |
| 15.6.1. Eventd | 292 |
| 15.6.2. Notifd | 292 |
| 15.6.3. Pollerd | 293 |
| 16. System Properties | 294 |
| 17. Ticketing | 295 |
| 17.1. JIRA Ticketing Plugin | 295 |
| 17.1.1. Setup | 295 |
| 17.1.2. Jira Commands | 296 |
| 17.1.3. Custom fields | 297 |
| 17.1.4. Troubleshooting | 301 |
| 17.2. Remedy Ticketing Plugin | 301 |
| 17.2.1. Remedy Product Overview | 301 |
| 17.2.2. Supported Remedy Product Versions | 301 |
| 17.2.3. Setup | 301 |
| 17.3. TSRM Ticketing Plugin | 304 |
| 17.3.1. Setup | 304 |
| 17.3.2. Mapping OpenNMS Ticket with TSRM Incident | 304 |
| 18. Enabling RMI | 306 |
| 18.1. Enabling RMI | 306 |
| 18.2. Enabling SSL | 307 |
| 18.3. Connecting to RMI over SSL | 307 |
| 19. Plugin Manager | 309 |
| 19.1. Plugin Manager UI panel | 309 |
| 19.2. Setting Karaf Instance Data | 311 |
| 19.3. Manually adding a managed <i>Karaf</i> instance | 313 |
| 19.4. Installed Plugins | 314 |

| 19.5. Available Plugins Server | |
|--|--|
| 19.6. Installing Available Plugins | |
| 19.7. Plugins Manifest | |
| 19.8. Installed Licences Panel | |
| 19.9. Adding a New Licence | |
| 19.10. Installing Internal Plugins | |
| 20. Internal Plugins 323 | |
| 20.1. Internal Plugins supplied with <i>OpenNMS Meridian</i> | |
| 20.2. Installing Plugins with the Karaf Consol | |
| 20.3. Alarm Change Notifier Plugin | |
| 20.4. Elasticsearch ReST plugin | |
| 20.4.1. Configuration | |
| 20.4.2. Loading Historical Events | |
| 20.4.3. Index Definitions | |
| 20.4.4. Viewing events using Kibana Sense | |
| 20.4.5. Mapping of Alarms and Events to Elasticsearch | |
| 21. Special Cases and Workarounds | |
| 21.1. Overriding SNMP Client Behavior | |
| | |

Chapter 1. Data Choices

The Data Choices module collects and publishes anonymous usage statistics to https://stats.opennms.org.

When a user with the Admin role logs into the system for the first time, they will be prompted as to whether or not they want to opt-in to publish these statistics. Statistics will only be published once an Administrator has opted-in.

Usage statistics can later be disabled by accessing the 'Data Choices' link in the 'Admin' menu.

When enabled, the following anonymous statistics will be collected and publish on system startup and every 24 hours after:

- System ID (a randomly generated UUID)
- OpenNMS Meridian Release
- OpenNMS Meridian Version
- OS Architecture
- OS Name
- OS Version
 - 1. Number of Alarms in the alarms table
 - 2. Number of Events in the events table
 - 3. Number of IP Interfaces in the ipinterface table
 - 4. Number of Nodes in the node table
 - 5. Number of Nodes, grouped by System OID

Chapter 2. User Management

Users are entities with login accounts in the *OpenNMS Meridian* system. Ideally each user corresponds to a person. An *OpenNMS Meridian User* represents an actor which may be granted permissions in the system by associating *Security Roles. OpenNMS Meridian* stores by default *User* information and credentials in a local embedded file based storage. Credentials and user details, e.g. contact information, descriptions or *Security Roles* can be managed through the *Admin Section* in the Web User Interface.

Beside local *Users*, external LDAP service and SSO can be configured, but are not scope in this section. The following paragraphs describe how to manage the embedded *User* and *Security Roles* in *OpenNMS Meridian*.

2.1. Users

Managing *Users* is done through the *Web User Interface* and requires to login as a *User* with administrative permissions. By default the *admin* user is used to initially create and modify *Users*. The *User*, *Password* and other detail descriptions are persisted in users.xml file. It is not required to restart *OpenNMS Meridian* when *User* attributes are changed.

In case administrative tasks should be delegated to an *User* the *Security Role* named *ROLE_ADMIN* can be assigned.



Don't delete the *admin* and *rtc* user. The *RTC* user is used for the communication of the Real-Time Console on the start page to calculate the node and service availability.



Change the default *admin* password to a secure password.

How to set a new password for any user

- 1. Login as a **User** with administrative permissions
- 2. Choose **Configure OpenNMS** from the user specific main navigation which is named as your login user name
- 3. Choose Configure Users, Groups and On-Call roles and select Configure Users
- 4. Click the **Modify** icon next to an existing *User* and select **Reset Password**
- 5. Set a new Password, Confirm Password and click OK
- 6. Click **Finish** to persist and apply the changes

How users can change their own password

- 1. Login with user name and old password
- 2. Choose **Change Password** from the user specific main navigation which is named as your login user name
- 3. Select Change Password
- 4. Identify yourself with the old password and set the new password and confirm

- 5. Click Submit
- 6. Logout and login with your new password

How to create or modify user

- 1. Login as a **User** with administrative permissions
- 2. Choose **Configure OpenNMS** from the user specific main navigation which is named as your login user name
- 3. Choose Configure Users, Groups and On-Call roles and select Configure Users
- 4. Use **Add new user** and type in a **login name** as *User ID* and a **Password** with confirmation or click **Modify** next to an existing *User*
- 5. *Optional*: Fill in detailed *User Information* to provide more context information around the new user in the system
- 6. Optional: Assign Security Roles to give or remove permissions in the system
- 7. Optional: Provide Notification Information which are used in Notification targets to send messages to the User
- 8. Optional: Set a schedule when a User should receive Notifications
- 9. Click **Finish** to persist and apply the changes



By default a new *User* has the *Security Role* similar to *ROLE_USER* assigned. Acknowledgment and working with *Alarms* and *Notifications* is possible. The *Configure OpenNMS* administration menu is not available.

How to delete existing user

- 1. Login as a **User** with administrative permissions
- 2. Choose **Configure OpenNMS** from the user specific main navigation which is named as your login user name
- 3. Choose Configure Users, Groups and On-Call roles and select Configure Users
- 4. Use the trash bin icon next to the *User* to delete
- 5. Confirm delete request with **OK**

2.2. Security Roles

A Security Roles is a set of permissions and can be assigned to an User. They regulate access to the Web User Interface and the ReST API to exchange monitoring and inventory information. In case of a distributed installation, the Remote Poller instances interact with OpenNMS Meridian and require specific permissions which are defined in the Security Role ROLE_REMOTING. The following Security Roles are available:

Table 1. Functions and existing system roles in OpenNMS Meridian

| Security Role Name | Description |
|-----------------------|---|
| anyone | In case the opennms-webapp-remoting package is installed, any user can download the <i>Java Webstart</i> installation package for the remote poller from http://opennms.server:8980/opennms-remoting/webstart/app.jnlp. |
| ROLE_ANONYMO US | Allows <i>HTTP OPTIONS</i> request to show allowed HTTP methods on a ReST resources and the login and logout page of the Web User Interface. |
| ROLE_ADMIN | Permissions to create, read, update and delete in the Web User Interface and the <i>ReST API</i> . |
| ROLE_ASSET_EDI TOR | Permissions to just update the asset records from nodes. |
| ROLE_DASHBOA RD | Allow users to just have access to the <i>Dashboard</i> . |
| ROLE_DELEGATE | Allows actions (such as acknowledging an alarm) to be performed on behalf of another user. |
| ROLE_JMX | Allows retrieving JMX metrics but does not allow executing MBeans of the <i>OpenNMS Meridian</i> JVM, even if they just return simple values. |
| ROLE_MOBILE | Allow user to use <i>OpenNMS COMPASS</i> mobile application to acknowledge <i>Alarms</i> and <i>Notifications</i> via the ReST API. |
| ROLE_PROVISIO N | Allow user to use the <i>Provisioning System</i> and configure <i>SNMP</i> in <i>OpenNMS Meridian</i> to access management information from devices. |
| ROLE_READONL Y | Limited to just read information in the Web User Interface and are no possibility to change <i>Alarm</i> states or <i>Notifications</i> . |
| ROLE_REMOTING | Permissions to allow access from a <i>Remote Poller</i> instance to exchange monitoring information. |
| ROLE_REST | Allow users interact with the whole ReST API of OpenNMS Meridian |
| ROLE_RTC | Exchange information with the <i>OpenNMS Meridian Real-Time Console</i> for availability calculations. |
| ROLE_USER | Default permissions of a new created user to interact with the Web User Interface which allow to escalate and acknowledge <i>Alarms</i> and <i>Notifications</i> . |

How to manage Security Roles for Users:

- 1. Login as a **User** with administrative permissions
- 2. Choose **Configure OpenNMS** from the user specific main navigation which is named as your login user name
- 3. Choose Configure Users, Groups and On-Call roles and select Configure Users
- 4. Modify an existing **User** by clicking the modify icon next to the **User**
- 5. Select the **Role** from **Available Roles** in the **Security Roles** section
- 6. Use **Add** and **Remove** to assign or remove the **Security Role** from the **User**
- 7. Click **Finish** to persist and apply the Changes

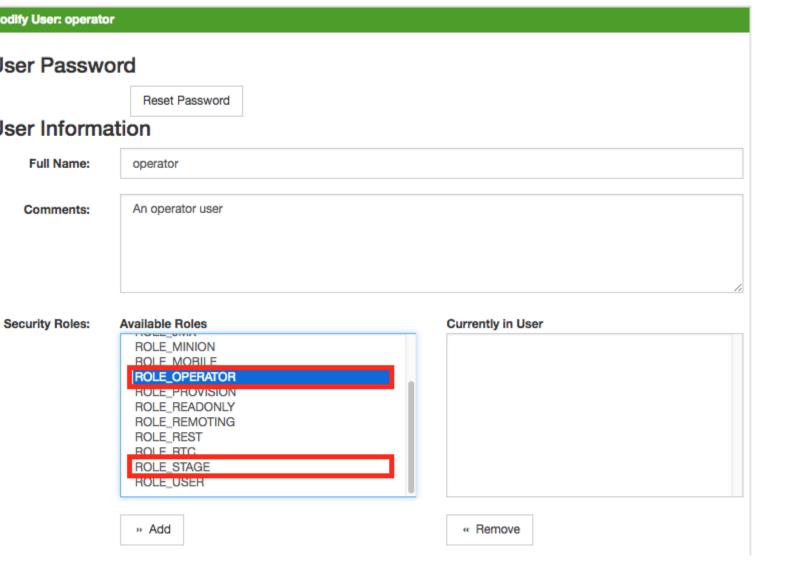
8. Logout and Login to apply the new Security Role settings

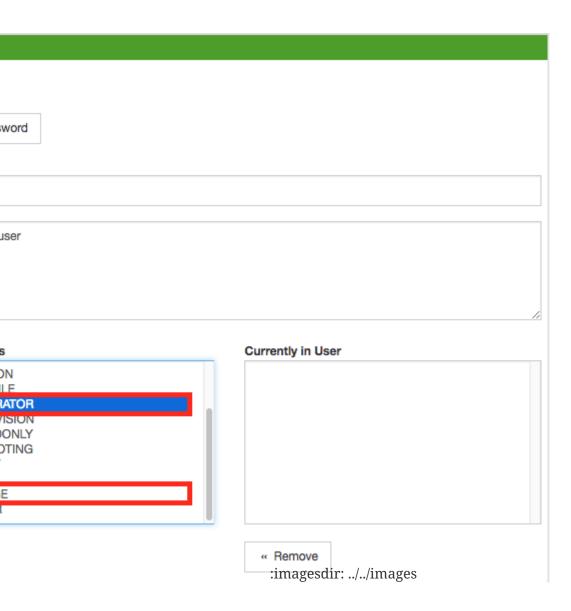
How to add custom roles

- Create a file called \$OPENNMS_HOME/etc/security-roles.properties.
- Add a property called roles, and for its value, a comma separated list of the custom roles, for example:

roles=operator,stage

• After following the procedure to associate the security roles with users, the new custom roles will be available as shown on the following image:





2.3. Web UI Pre-Authentication

It is possible to configure *OpenNMS Meridian* to run behind a proxy that provides authentication, and then pass the pre-authenticated user to the *OpenNMS Meridian* webapp using a header.

The pre-authentication configuration is defined in <code>\$OPENNMS_HOME/jetty-webapps/opennms/WEB-INF/spring-security.d/header-preauth.xml</code>. This file is automatically included in the Spring Security context, but is not enabled by default.



DO NOT configure *OpenNMS Meridian* in this manner unless you are certain the web UI is only accessible to the proxy and not to end-users. Otherwise, malicious attackers can craft queries that include the pre-authentication header and get full control of the web UI and ReST APIs.

2.3.1. Enabling Pre-Authentication

Edit the header-preauth.xml file, and set the enabled property:

```
<beans:property name="enabled" value="true" />
```

2.3.2. Configuring Pre-Authentication

There are a number of other properties that can be set to change the behavior of the preauthentication plugin.

| Property | Description | Default |
|-------------------|--|---------------|
| enabled | Whether the pre-authentication plugin is active. | false |
| failOnError | If true, disallow login if the header is not set or the user does not exist. If false, fall through to other mechanisms (basic auth, form login, etc.) | false |
| userHeader | The HTTP header that will specify the user to authenticate as. | X-Remote-User |
| credentialsHeader | A comma-separated list of additional credentials (roles) the user should have. | |

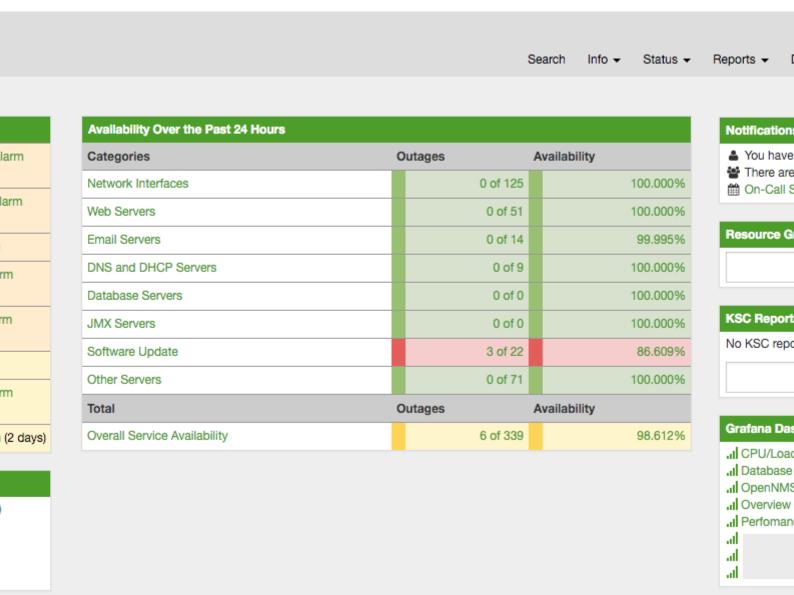
Chapter 3. Administrative Webinterface

3.1. Grafana Dashboard Box

Grafana provides an API key which gives access for 3rd party application like *OpenNMS Meridian*. The *Grafana Dashboard Box* on the start page shows dashboards related to *OpenNMS Meridian*. To filter relevant dashboards, you can use a *tag* for dashboards and make them accessible. If no *tag* is provided all dashboards from *Grafana* will be shown.

The feature is by default deactivated and is configured through opennms.properties. Please note that this feature works with the *Grafana API v2.5.0*.

Quick access to Grafana dashboards from the OpenNMS Meridian start page



Notification

You have

There are

Resource G

KSC Report

No KSC repo

Grafana Das

III CPU/Load
III Database
III OpenNMS
III Overview
III Perfoman

al al al

| Categories | Outages | Availal | bility |
|------------------------------|---------|----------|----------|
| Network Interfaces | | 0 of 125 | 100.000% |
| Web Servers | | 0 of 51 | 100.000% |
| Email Servers | | 0 of 14 | 99.995% |
| DNS and DHCP Servers | | 0 of 9 | 100.000% |
| Database Servers | | 0 of 0 | 100.000% |
| JMX Servers | | 0 of 0 | 100.000% |
| Software Update | | 3 of 22 | 86.609% |
| Other Servers | | 0 of 71 | 100.000% |
| Total | Outages | Availal | bility |
| Overall Service Availability | | 6 of 339 | 98.6129 |

Table 2. Grafana Dashboard configuration properties

larm

larm

rm

m

rm

(2 days)

| Name | Туре | Description | Defau lt |
|-------------------------------------|-------------|---|---------------|
| org.opennms.grafanaBox.s how | Boole an | This setting controls whether a grafana box showing the available dashboards is placed on the landing page. The two valid options for this are true or false. | false |
| org.opennms.grafanaBox.h ostname | Strin g | If the box is enabled you also need to specify hostname of the <i>Grafana</i> server | localh ost |
| org.opennms.grafanaBox.p ort | Integ er | The port of the <i>Grafana</i> server ReST API | 3000 |
| org.opennms.grafanaBox.a piKey | Strin g | The API key is needed for the ReST calls to work | |

| Name | Туре | Description | Defau lt |
|--|-------------|--|-------------|
| org.opennms.grafanaBox.t | Strin g | When a <i>tag</i> is specified only dashboards with this given <i>tag</i> will be displayed. When no <i>tag</i> is given all dashboards will be displayed | |
| org.opennms.grafanaBox.protocol | Strin g | The protocol for the ReST call can also be specified | http |
| org.opennms.grafanaBox.connectionTimeout | Integ er | Timeout in milliseconds for getting information from the <i>Grafana</i> server | 500 |
| org.opennms.grafanaBox.s oTimeout | Integ er | | 500 |



If you have *Grafana* behind a proxy it is important the org.opennms.grafanaBox.hostname is reachable. This host name is used to generate links to the *Grafana* dashboards.

The process to generate an *Grafana API Key* can be found in the HTTP API documentation. Copy the API Key to opennms.properties as org.opennms.grafanaBox.apiKey.

3.2. Operator Board

In a network operation center *(NOC)* the *Ops Board* can be used to visualize monitoring information. The monitoring information for various use-cases are arranged in configurable *Dashlets*. To address different user groups it is possible to create multiple *Ops Boards*.

There are two visualisation components to display *Dashlets*:

- Ops Panel: Shows multiple Dashlets on one screen, e.g. on a NOC operators workstation
- Ops Board: Shows one Dashlet at a time in rotation, e.g. for a screen wall in a NOC

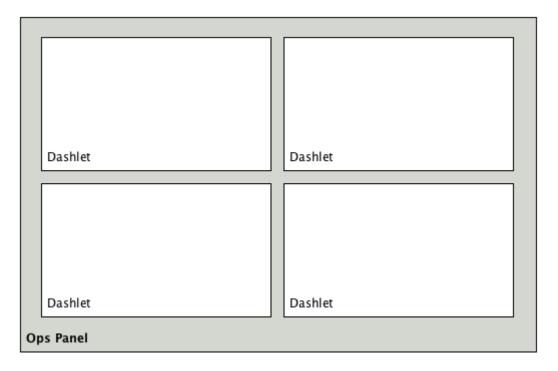


Figure 1. Concept of Dashlets displayed in Ops Panel

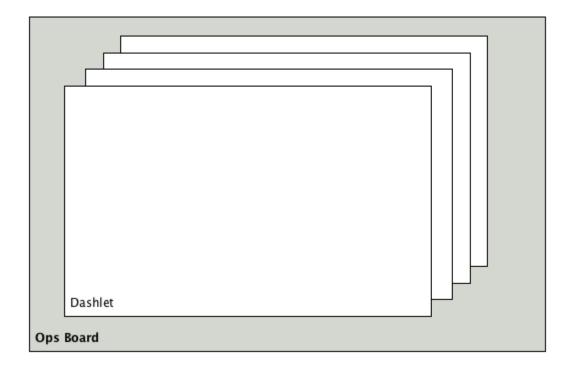


Figure 2. Concept to show Dashlets in rotation on the Ops Board

3.2.1. Configuration

To create and configure *Ops Boards* administration permissions are required. The configuration section is in admin area of OpenNMS Meridian and named *Ops Board Config Web Ui*.

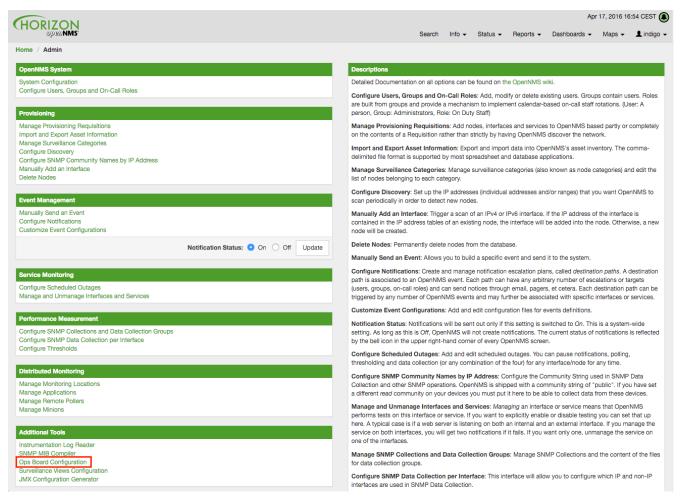


Figure 3. Navigation to the Ops Board configuration

Create or modify *Ops Boards* is described in the following screenshot.

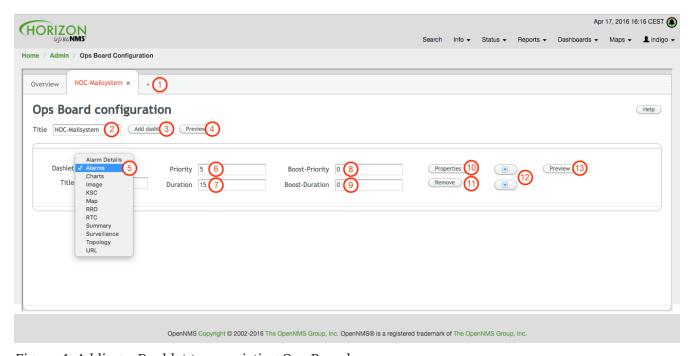


Figure 4. Adding a Dashlet to an existing Ops Board

- 1. Create a new *Ops Board* to organize and arrange different *Dashlets*
- 2. The name to identify the *Ops Board*
- 3. Add a Dashlet to show OpenNMS Meridian monitoring information

- 4. Show a preview of the whole Ops Board
- 5. List of available Dashlets
- 6. *Priority* for this *Dashlet* in *Ops Board* rotation, lower priority means it will be displayed more often
- 7. Duration in seconds for this Dashlet in the Ops Board rotation
- 8. Change *Priority* if the *Dashlet* is in alert state, this is optional and maybe not available in all *Dashlets*
- 9. Change *Duration* if the *Dashlet* is in alert state, it is optional and maybe not available in all *Dashlets*
- 10. Configuration properties for this Dashlet
- 11. Remove this Dashlet from the Ops Board
- 12. Order Dashlets for the rotation on the Ops Board and the tile view in the Ops Panel
- 13. Show a preview for the whole *Ops Board*

The configured *Ops Board* can be used by navigating in the main menu to *Dashboard* \rightarrow *Ops Board*.



Figure 5. Navigation to use the Ops Board

3.2.2. Dashlets

Visualization of information is implemented in *Dashlets*. The different *Dashlets* are described in this section with all available configuration parameter.

To allow filter information the *Dashlet* can be configured with a generic Criteria Builder.

Alarm Details

This Alarm-Details Dashlet shows a table with alarms and some detailed information.

Table 3. Information of the alarms

| Field | Description |
|-----------------|--|
| Alarm ID | OpenNMS Meridian ID for the alarm |
| Severity | Alarm severity (Cleared, Indeterminate, Normal, Warning, Minor, Major, Critical) |
| Node label | Node label of the node where the alarm occurred |
| Alarm count | Alarm count based on reduction key for deduplication |
| Last Event Time | Last time the alarm occurred |
| Log Message | Reason and detailed log message of the alarm |

The *Alarm Details Dashlet* can be configured with the following parameters.

| Boost support | Boosted Severity | |
|---------------|------------------|--|
| Configuration | Criteria Builder | |

Alarms

This *Alarms Dashlet* shows a table with a short alarm description.

Table 4. Information of the alarm

| Field | Description |
|------------|--|
| Time | Absolute time since the alarm appeared |
| Node label | Node label of the node where the alarm occurred |
| UEI | OpenNMS Meridian <i>Unique Event Identifier</i> for this alarm |

The *Alarms Dashlet* can be configured with the following parameters.

| Boost support | Boosted Severity |
|---------------|------------------|
| Configuration | Criteria Builder |

Charts

This Dashlet displays an existing Chart.

| Boost support | false |
|-----------------|--|
| Chart | Name of the existing chart to display |
| Maximize Width | Rescale the image to fill display width |
| Maximize Height | Rescale the image to fill display height |

Grafana

This *Dashlet* shows a *Grafana Dashboard* for a given time range. The *Grafana Dashboard Box* configuration defined in the opennms.properties file is used to access the *Grafana* instance.

| Boost support | false |
|---------------|--|
| title | Title of the Grafana dashboard to be displayed |
| uri | URI to the Grafana Dashboard to be displayed |
| from | Start of time range |
| to | End of time range |

Image

This Dashlet displays an image by a given URL.

| Boost support | false | |
|----------------|---|--|
| imageUrl | URL with the location of the image to show in this <i>Dashlet</i> | |
| maximizeHeight | Rescale the image to fill display width | |
| maximizeWidth | Rescale the image to fill display height | |

KSC

This *Dashlet* shows an existing KSC report. The view is exact the same as the KSC report is build regarding order, columns and time spans.

| Boost support | false |
|---------------|---|
| KSC-Report | Name of the KSC report to show in this <i>Dashlet</i> |

Map

This Dashlet displays the geographical map.

| Boost support | false |
|------------------|---|
| search | Predefined search for a subset of nodes shown in the geographical map in this Dashlet |

RRD

This *Dashlet* shows one or multiple RRD graphs. It is possible to arrange and order the RRD graphs in multiple columns and rows. All RRD graphs are normalized with a given width and height.

| Boost support | false |
|---------------|--------------------------------------|
| Columns | Number of columns within the Dashlet |

| Rows | Number of rows with the <i>Dashlet</i> | |
|-----------------|--|--|
| KSC Report | Import RRD graphs from an existing KSC report and re-arrange them. | |
| Graph Width | Generic width for all RRD graphs in this Dashlet | |
| Graph Height | Generic height for all RRD graphs in this Dashlet | |
| Timeframe value | Number of the given Timeframe type | |
| Timeframe type | Minute, Hour, Day, Week, Month and Year for all RRD graphs | |

RTC

This Dashlet shows the configured SLA categories from the OpenNMS Meridian start page.

| Boost support | false |
|---------------|-------|
| _ | - |

Summary

This *Dashlet* shows a trend of incoming alarms in given time frame.

| Boost support | Boosted Severity |
|---------------|--|
| timeslot | Time slot in seconds to evaluate the trend for alarms by severity and <i>UEI</i> . |

Surveillance

This Dashlet shows a given Surveillance View.

| Boost support | false |
|---------------|--|
| viewName | Name of the configured Surveillance View |

Topology

This *Dashlet* shows a Topology Map. The *Topology Map* can be configured with the following parameter.

| Boost support | false |
|---------------|--|
| focusNodes | Which node(s) is in focus for the topology |
| provider | Which topology should be displayed, e.g. Linkd, VMware |
| szl | Set the zoom level for the topology |

URL

This *Dashlet* shows the content of a web page or other web application, e.g. other monitoring systems by a given URL.

| Boost support | false |
|---------------|---|
| password | Optional password if a basic authentication is required |
| url | URL to the web application or web page |
| username | Optional username if a basic authentication is required |

3.2.3. Boosting Dashlet

The behavior to boost a *Dashlet* describes the behavior of a *Dashlet* showing critical monitoring information. It can raise the priority in the *Ops Board* rotation to indicate a problem. This behavior can be configured with the configuration parameter *Boost Priority* and *Boost Duration*. These to configuration parameter effect the behavior on the *Ops Board* in rotation.

- Boost Priority: Absolute priority of the Dashlet with critical monitoring information.
- Boost Duration: Absolute duration in seconds of the Dashlet with critical monitoring information.

3.2.4. Criteria Builder

The *Criteria Builder* is a generic component to filter information of a *Dashlet*. Some *Dashlets* use this component to filter the shown information on a *Dashlet* for certain use case. It is possible to combine multiple *Criteria* to display just a subset of information in a given *Dashlet*.

Table 5. Generic Criteria Builder configuration possibilities

| Restrict ion | Property | Value 1 | Value 2 | Description |
|--------------|-----------------------|------------|------------|--|
| Asc | - | - | - | ascending order |
| Desc | - | - | - | descending order |
| Between | database attribute | String | String | Subset of data between value 1 and value 2 |
| Contains | database attribute | String | - | Select all data which contains a given text string in a given database attribute |
| Distinct | database attribute | - | - | Select a single instance |
| Eq | database attribute | String | - | Select data where attribute equals (==) a given text string |
| Ge | database attribute | String | - | Select data where attribute is greater equals than (>=) a given text value |
| Gt | database attribute | String | - | Select data where attribute is greater than (>) a given text value |
| Ilike | database attribute | String | - | unknown |

| Restrict ion | Property | Value 1 | Value 2 | Description |
|---------------|-----------------------|-------------|------------|--|
| In | database attribute | String | - | unknown |
| Iplike | database attribute | String | - | Select data where attribute matches an given IPLIKE expression |
| IsNull | database attribute | - | - | Select data where attribute is null |
| IsNotNul l | database attribute | - | - | Select data where attribute is not null |
| IsNotNul l | database attribute | - | - | Select data where attribute is not null |
| Le | database attribute | String | - | Select data where attribute is less equals than (←) a given text value |
| Lt | database attribute | String | - | Select data where attribute is less than (<) a given text value |
| Le | database attribute | String | - | Select data where attribute is less equals than (←) a given text value |
| Like | database attribute | String | - | Select data where attribute is like a given text value similar to SQL like |
| Limit | - | Intege r | - | Limit the result set by a given number |
| Ne | database attribute | String | - | Select data where attribute is not equals (!=) a given text value |
| Not | database attribute | String | - | unknown difference between Ne |
| OrderBy | database attribute | - | - | Order the result set by a given attribute |

3.3. JMX Configuration Generator

OpenNMS Meridian implements the *JMX* protocol to collect long term performance data for *Java* applications. There are a huge variety of metrics available and administrators have to select which information should be collected. The *JMX Configuration Generator Tools* is build to help generating valid complex *JMX* data collection configuration and *RRD graph* definitions for *OpenNMS Meridian*.

This tool is available as CLI and a web based version.

3.3.1. Web based utility

Complex *JMX* data collection configurations can be generated from a web based tool. It collects all available *MBean Attributes* or *Composite Data Attributes* from a *JMX* enabled Java application.

The workflow of the tool is:

- 1. Connect with JMX or JMXMP against a MBean Server provided of a Java application
- 2. Retrieve all *MBean* and *Composite Data* from the application
- 3. Select specific *MBeans* and *Composite Data* objects which should be collected by *OpenNMS Meridian*
- 4. Generate *JMX Collectd* configuration file and *RRD graph* definitions for *OpenNMS Meridian* as downloadable archive

The following connection settings are supported:

- Ability to connect to MBean Server with RMI based IMX
- Authentication credentials for JMX connection
- Optional: JMXMP connection

The web based configuration tool can be used in the *OpenNMS Meridian Web Application* in administration section $Admin \rightarrow JMX$ Configuration Generator.

Configure JMX Connection

At the beginning the connection to an *MBean Server* of a *Java* application has to be configured.

| ? 1. Servi | ce Configuration / 2. MBeans Configuration / 3. OpenNMS Configuration |
|----------------|---|
| Service name * | OpenNMS-JVM |
| Host* | localhost |
| Port* | 18980 |
| | Authentication |
| | ✓ Skip JVM MBeans |
| | Skip non-number values |
| | |
| | |
| > | |

Figure 6. JMX connection configuration window

- Service name: The name of the service to bind the JMX data collection for Collectd
- *Host*: IP address or *FQDN* connecting to the *MBean Server* to load *MBeans* and *Composite Data* into the generation tool
- Port: Port to connect to the MBean Server
- Authentication: Enable / Disable authentication for JMX connection with username and

password

- Skip non-number values: Skip attributes with non-number values
- JMXMP: Enable / Disable JMX Messaging Protocol instead of using JMX over RMI

By clicking the arrow (>) the *MBeans* and *Composite Data* will be retrieved with the given connection settings. The data is loaded into the *MBeans Configuration* screen which allows to select metrics for the data collection configuration.

Select MBeans and Composite

The *MBeans Configuration* section is used to assign the *MBean* and *Composite Data attributes* to *RRD domain* specific data types and data source names.

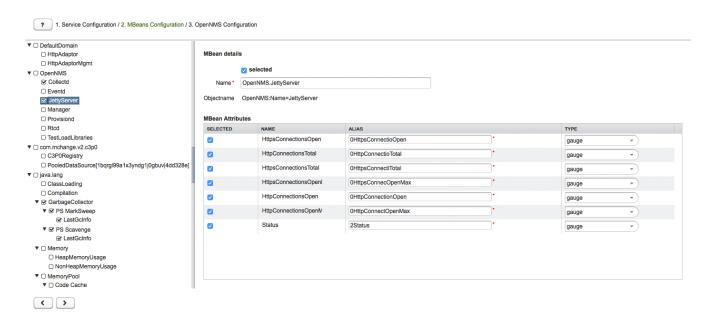


Figure 7. Select MBeans or Composite Data for OpenNMS Meridian data collection

The left sidebar shows the tree with the *JMX Domain*, *MBeans* and *Composite Data* hierarchy retrieved from the *MBean Server*. To select or deselect all attributes use *Mouse right click* \rightarrow *select/deselect*.

The right panel shows the *MBean Attributes* with the *RRD* specific mapping and allows to select or deselect specific *MBean Attributes* or *Composite Data Attributes* for the data collection configuration.

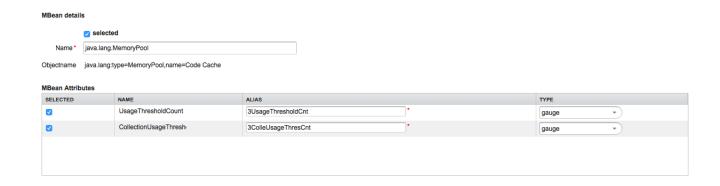


Figure 8. Configure MBean attributes for data collection configuration



Figure 9. Configure Composite attributes for data collection configuration

- MBean Name or Composite Alias: Identifies the MBean or the Composite Data object
- *Selected*: Enable/Disable the *MBean attribute* or *Composite Member* to be included in the data collection configuration
- Name: Name of the MBean attribute or Composite Member
- Alias: the data source name for persisting measurements in RRD or JRobin file
- Type: Gauge or Counter data type for persisting measurements in RRD or JRobin file

The *MBean Name*, *Composite Alias* and *Name* are validated against special characters. For the *Alias* inputs are validated to be not longer then 19 characters and have to be unique in the data collection configuration.

Download and include configuration

The last step is generating the following configuration files for *OpenNMS Meridian*:

- collectd-configuration.xml: Generated sample configuration assigned to a service with a matching data collection group
- *jmx-datacollection-config.xml*: Generated *JMX* data collection configuration with the selected *MBeans* and *Composite Data*
- snmp-graph.properties: Generated default RRD graph definition files for all selected metrics

The content of the configuration files can be copy & pasted or can be downloaded as *ZIP archive*.



If the content of the configuration file exceeds 2,500 lines, the files can only be downloaded as *ZIP archive*.

3.3.2. CLI based utility

The command line *(CLI)* based tool is not installed by default. It is available as *Debian* and *RPM* package in the official repositories.

Installation

RHEL based installation with Yum

```
yum install opennms-jmx-config-generator
```

Debian based installation with apt

```
apt-get install opennms-jmx-config-generator
```

Installation from source

It is required to have the *Java 8 Development Kit* with *Apache Maven* installed. The mvn binary has to be in the path environment. After cloning the repository you have to enter the source folder and compile an executable *JAR*.

```
cd opennms/features/jmx-config-generator
mvn package
```

Inside the newly created target folder a file named jmxconfiggenerator-<VERSION>-onejar.jar is present. This file can be invoked by:

```
java -jar target/jmxconfiggenerator-2017.1.6-onejar.jar
```

Usage

After installing the the *JMX Config Generator* the tool's wrapper script is located in the \${OPENNMS_HOME}/bin directory.

```
$ cd /path/to/opennms/bin
```

\$./jmx-config-generator



When invoked without parameters the usage and help information is printed.

The *JMX Config Generator* uses sub-commands for the different configuration generation tasks. Each of these sub-commands provide different options and parameters. The command line tool accepts the following sub-commands.

| Sub- command | Description |
|--------------------|---|
| query | Queries a MBean Server for certain MBeans and attributes. |
| generate- conf | Generates a valid jmx-datacollection-config.xml file. |
| generate- graph | Generates a <i>RRD</i> graph definition file with matching graph definitions for a given jmx-datacollection-config.xml. |

The following global options are available in each of the sub-commands of the tool:

| Option/Argume nt | Description | Defaul t |
|---------------------|--|-------------|
| -h (help) | Show help and usage information. | false |
| -v (verbose) | Enables verbose mode for debugging purposes. | false |

Sub-command: query

This sub-command is used to query a *MBean Server* for it's available *MBean* objects. The following example queries the server myserver with the credentials myusername/mypassword on port 7199 for *MBean objects* in the java.lang domain.

```
./jmx-config-generator query --host myserver --username myusername --password
mypassword --port 7199 "java.lang:*"
java.lang:type=ClassLoading
    description: Information on the management interface of the MBean
    class name: sun.management.ClassLoadingImpl
    attributes: (5/5)
        TotalLoadedClassCount
            id: java.lang:type=ClassLoading:TotalLoadedClassCount
            description: TotalLoadedClassCount
            type: long
            isReadable: true
            isWritable: false
            isIs: false
        LoadedClassCount
            id: java.lang:type=ClassLoading:LoadedClassCount
            description: LoadedClassCount
            type: int
            isReadable: true
            isWritable: false
            isIs: false
<output omitted>
```

The following command line options are available for the *query* sub-command.

| Option/Argument | Description | Defa ult |
|--------------------------------------|--|-------------|
| <filter criteria=""></filter> | A filter criteria to query the MBean Server for. The format is <objectname>[:attribute name]. The <objectname> accepts the default JMX object name pattern to identify the MBeans to be retrieved. If null all domains are shown. If no key properties are specified, the domain's MBeans are retrieved. To execute for certain attributes, you have to add :<attribute name="">. The <attribute name=""> accepts regular expressions. When multiple <filter criteria=""> are provided they are OR concatenated.</filter></attribute></attribute></objectname></objectname> | - |
| host <host></host> | Hostname or IP address of the remote <i>JMX</i> host. | - |
| ids-only | Only show the ids of the attributes. | false |
| ignore <filter criteria=""></filter> | Set <filter criteria=""> to ignore while running.</filter> | - |
| include-values | Include attribute values. | false |
| jmxmp | Use JMXMP and not JMX over RMI. | false |
| password <password></password> | Password for <i>JMX</i> authentication. | - |
| port <port></port> | Port of <i>JMX</i> service. | - |
| show-domains | Only lists the available domains. | true |
| show-empty | Includes <i>MBeans</i> , even if they do not have attributes. Either due to the <filter criteria=""> or while there are none.</filter> | false |
| url <url></url> | <pre>Custom connection URL <hostname>:<port> service:jmx:<pre>cyrotocol>:<sap> service:jmx:remoting-jmx://<hostname>:<port></port></hostname></sap></pre></port></hostname></pre> | - |
| username <username></username> | Username for <i>JMX</i> authentication. | - |
| -h (help) | Show help and usage information. | false |
| -v (verbose) | Enables verbose mode for debugging purposes. | false |

Sub-command: generate-conf

This sub-command can be used to generate a valid <code>jmx-datacollection-config.xml</code> for a given set of <code>MBean objects</code> queried from a <code>MBean Server</code>.

The following example generate a configuration file myconfig.xml for *MBean* objects in the java.lang domain of the server myserver on port 7199 with the credentials myusername/mypassword. You have to define either an *URL* or a hostname and port to connect to a *JMX* server.

jmx-config-generator generate-conf --host myserver --username myusername --password
mypassword --port 7199 "java.lang:*" --output myconfig.xml
Dictionary entries loaded: '18'

The following options are available for the *generate-conf* sub-command.

| Option/Argume nt | Description | Default |
|-----------------------------------|--|----------------|
| <attribute id=""></attribute> | A list of attribute Ids to be included for the generation of the configuration file. | - |
| dictionary <file></file> | Path to a dictionary file for replacing attribute names and part of <i>MBean</i> attributes. The file should have for each line a replacement, e.g. Auxillary:Auxil. | - |
| host <host></host> | Hostname or IP address of JMX host. | - |
| jmxmp | Use JMXMP and not JMX over RMI. | false |
| output <file></file> | Output filename to write generated jmx-datacollection-config.xml. | - |
| password <password></password> | Password for <i>JMX</i> authentication. | - |
| port <port></port> | Port of JMX service | - |
| print -dictionary | Prints the used dictionary to <i>STDOUT</i> . May be used withdictionary | false |
| service <value></value> | The Service Name used as JMX data collection name. | anyserv ice |
| skipDefaultVM | Skip default JavaVM Beans. | false |
| skipNonNumber | Skip attributes with non-number values | false |
| url <url></url> | Custom connection URL <hostname>:<port> service:jmx:<protocol>:<sap> service:jmx:remoting-jmx://<hostname>:<port></port></hostname></sap></protocol></port></hostname> | - |
| username <username></username> | Username for <i>JMX</i> authentication | - |
| -h (help) | Show help and usage information. | false |
| -v (verbose) | Enables verbose mode for debugging purposes. | false |



The option <code>--skipDefaultVM</code> offers the ability to ignore the <code>MBeans</code> provided as standard by the <code>JVM</code> and just create configurations for the <code>MBeans</code> provided by the <code>Java Application</code> itself. This is particularly useful if an optimized configuration for the <code>JVM</code> already exists. If the <code>--skipDefaultVM</code> option is not set the generated configuration will include the <code>MBeans</code> of the <code>JVM</code> and the <code>MBeans</code> of the <code>Java Application</code>.



Check the file and see if there are alias names with more than 19 characters. This errors are marked with NAME_CRASH_AS_19_CHAR_VALUE

Sub-command: generate-graph

This sub-command generates a *RRD* graph definition file for a given configuration file. The following example generates a graph definition file mygraph.properties using the configuration in file myconfig.xml.

```
./jmx-config-generator generate-graph --input myconfig.xml --output mygraph.properties reports=java.lang.ClassLoading.MBeanReport, \
java.lang.ClassLoading.OTotalLoadeClassCnt.AttributeReport, \
java.lang.ClassLoading.OLoadedClassCnt.AttributeReport, \
java.lang.ClassLoading.OUnloadedClassCnt.AttributeReport, \
java.lang.Compilation.MBeanReport, \
<output omitted>
```

The following options are available for this sub-command.

| Option/Argument | Description | Defa ult |
|--|---|-------------|
| <pre>input <jmx- datacollection.xml=""></jmx-></pre> | Configuration file to use as input to generate the graph properties file | - |
| output <file></file> | Output filename for the generated graph properties file. | - |
| print-template | Prints the default template. | false |
| template <file></file> | Template file using <i>Apache Velocity</i> template engine to be used to generate the graph properties. | - |
| -h (help) | Show help and usage information. | false |
| -v (verbose) | Enables verbose mode for debugging purposes. | false |

Graph Templates

The *JMX Config Generator* uses a template file to generate the graphs. It is possible to use a user-defined template. The option --template followed by a file lets the *JMX Config Generator* use the external template file as base for the graph generation. The following example illustrates how a custom template mytemplate.vm is used to generate the graph definition file mygraph.properties using the configuration in file myconfig.xml.

```
./jmx-config-generator generate-graph --input myconfig.xml --output mygraph.properties --template mytemplate.vm
```

The template file has to be an *Apache Velocity* template. The following sample represents the template that is used by default:

```
reports=#foreach( $report in $reportsList )
${report.id}#if( $foreach.hasNext ), \
#end
#end
#foreach( $report in $reportsBody )
#[[##]]# $report.id
report.${report.id}.name=${report.name}
report.${report.id}.columns=${report.graphResources}
report.${report.id}.type=interfaceSnmp
report.${report.id}.command=--title="${report.title}" \
--vertical-label="${report.verticalLabel}" \
#foreach($graph in $report.graphs )
DEF:${graph.id}={rrd${foreach.count}}:${graph.resourceName}:AVERAGE \
AREA:${graph.id}#${graph.coloreB} \
LINE2:${graph.id}#${graph.coloreA}:"${graph.description}" \
GPRINT:${graph.id}:AVERAGE:" Avg \\: %8.21f %s" \
GPRINT:${graph.id}:MIN:" Min \\: %8.21f %s" \
GPRINT:${graph.id}:MAX:" Max \\: %8.21f %s\\n" \
#end
#end
```

The *JMX Config Generator* generates different types of graphs from the jmx-datacollection-config.xml. The different types are listed below:

| Туре | Description |
|------------------------------|---|
| AttributeReport | For each attribute of any <i>MBean</i> a graph will be generated. Composite attributes will be ignored. |
| MbeanReport | For each <i>MBean</i> a combined graph with all attributes of the <i>MBeans</i> is generated. Composite attributes will be ignored. |
| CompositeReport | For each composite attribute of every <i>MBean</i> a graph is generated. |
| CompositeAttribute Report | For each composite member of every <i>MBean</i> a combined graph with all composite attributes is generated. |

3.4. Heatmap

The *Heatmap* can be either be used to display unacknowledged alarms or to display ongoing outages of nodes. Each of this visualizations can be applied on categories, foreign sources or services of nodes. The sizing of an entity is calculated by counting the services inside the entity. Thus, a node with fewer services will appear in a smaller box than a node with more services.

The feature is by default deactivated and is configured through opennms.properties.





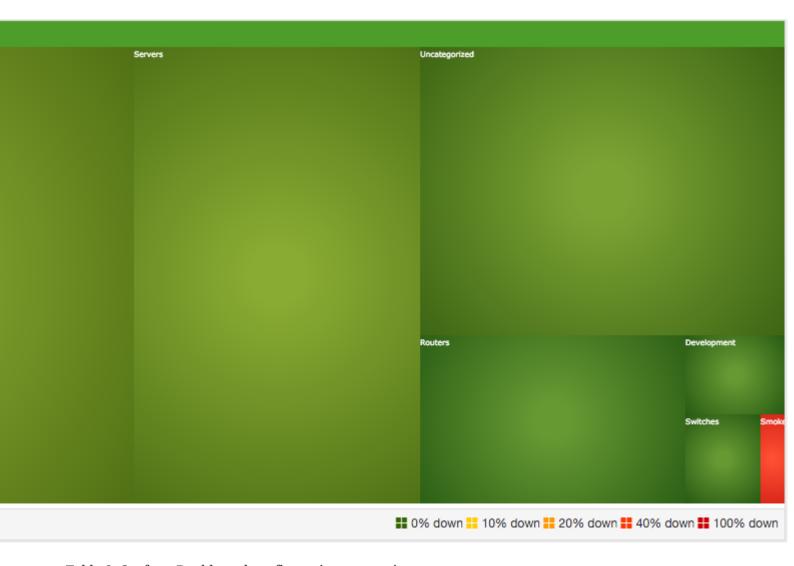


Table 6. Grafana Dashboard configuration properties

| Name | Туре | Description | Default |
|--|------------|---|------------|
| org.opennms.heatmap.d efaultMode | Strin g | There exist two options for using the heatmap: alarms and outages. This option configures which are displayed per default. | alarms |
| org.opennms.heatmap.d efaultHeatmap | Strin g | This option defines which <i>Heatmap</i> is displayed by default. Valid options are categories, foreignSources and monitoredServices. | categories |
| org.opennms.heatmap.c ategoryFilter | Strin g | The following option is used to filter for categories to be displayed in the Heatmap. This option uses the Java regular expression syntax. The default is .* so all categories will be displayed. | .* |

| Name | Туре | Description | Default |
|---|-------------|---|---------------------------|
| org.opennms.heatmap.f oreignSourceFilter | Strin g | The following option is used to filter for foreign sources to be displayed in the <i>Heatmap</i> . This option uses the Java regular expression syntax. The default is .* so all foreign sources will be displayed. | .* |
| org.opennms.heatmap.s erviceFilter | Strin g | The following option is used to filter for services to be displayed in the <i>Heatmap</i> . This option uses the Java regular expression syntax. The default is .* so all services will be displayed. | .* |
| org.opennms.heatmap.o nlyUnacknowledged | Bool ean | This option configures whether only unacknowledged alarms will be taken into account when generating the alarm-based version of the <i>Heatmap</i> . | false |
| org.opennms.web.conso le.centerUrl | Strin g | You can also place the <i>Heatmap</i> on the landing page by setting this option to /heatmap/heatmap-box.jsp. | /surveillanc e-box.jsp |



You can use negative lookahead expressions for excluding categories you wish not to be displayed in the heatmap, e.g. by using an expression like $^{(?!XY).*}$ you can filter out entities with names starting with XY.

3.5. Trend

The *Trend* feature allows to display small inline charts of database-based statistics. These chart are accessible in the *Status* menu of the *OpenNMS'* web application. Furthermore it is also possible to configure these charts to be displayed on the *OpenNMS'* landing page. To achieve this alter the org.opennms.web.console.centerUrl property to also include the entry /trend/trend-box.htm.

Trend chart structure





These charts can be configured and defined in the trend-configuration.xml file in your *OpenNMS'* etc directory. The following sample defines a *Trend* chart for displaying nodes with ongoing outages.

```
<trend-definition name="nodes">
     <title>Nodes</title> ①
     <subtitle>w/ Outages</subtitle> ②
     <visible>true</visible> 3
     <icon>glyphicon-fire</icon> 4
     <trend-attributes> (5)
         <trend-attribute key="sparkWidth" value="100%"/>
         <trend-attribute key="sparkHeight" value="35"/>
         <trend-attribute key="sparkChartRangeMin" value="0"/>
         <trend-attribute key="sparkLineColor" value="white"/>
         <trend-attribute key="sparkLineWidth" value="1.5"/>
         <trend-attribute key="sparkFillColor" value="#88BB55"/>
         <trend-attribute key="sparkSpotColor" value="white"/>
         <trend-attribute key="sparkMinSpotColor" value="white"/>
         <trend-attribute key="sparkMaxSpotColor" value="white"/>
         <trend-attribute key="sparkSpotRadius" value="3"/>
         <trend-attribute key="sparkHighlightSpotColor" value="white"/>
         <trend-attribute key="sparkHighlightLineColor" value="white"/>
     </trend-attributes>
     <descriptionLink>outage/list.htm?outtype=current</descriptionLink> 6
     <description>${intValue[23]} NODES WITH OUTAGE(S)</description> ⑦
     <query> 8
         <![CDATA[
             select (
                 select
                     count(distinct nodeid)
                 from
                     outages o, events e
                 where
                     e.eventid = o.svclosteventid
                     and iflostservice < E
                     and (ifregainedservice is null
                         or ifregainedservice > E)
             ) from (
                 select
                     now() - interval '1 hour' * (0 + 1) AS S,
                     now() - interval '1 hour' * 0 as E
                     generate_series(0, 23) as 0
             ) I order by S;
         11>
     </query>
 </trend-definition>
```

- 1 title of the *Trend* chart, see below for supported variable substitutions
- ② subtitle of the *Trend* chart, see below for supported variable substitutions
- 3 defines whether the chart is visible by default

- 4 icon for the chart, see Glyphicons for viable options
- 5 options for inline chart, see jQuery Sparklines for viable options
- 6 the description link
- ① the description text, see below for supported variable substitutions
- 8 the SQL statement for querying the chart's values
 - •

Don't forget to limit the SQL query's return values!

It is possible to use values or aggregated values in the title, subtitle and description fields. The following table describes the available variable substitutions.

Table 7. Variables usable in definition's title, subtitle and description fields

| Name | Туре | Description |
|---------------------------------|---------|--|
| \${intMax} | Integer | integer maximum value |
| \${doubleMax} | Double | maximum value |
| \${intMin} | Integer | integer minimum value |
| \${doubleMin} | Double | minimum value |
| \${intAvg} | Integer | integer average value |
| \${doubleAvg} | Double | average value |
| \${intSum} | Integer | integer sum of values |
| \${doubleSum} | Double | sum of value |
| \${intValue[]} | Integer | array of integer result values for the given SQL query |
| \${doubleValue[]} | Double | array of result values for the given SQL query |
| <pre>\${intValueChange[]}</pre> | Integer | array of integer value changes for the given SQL query |
| \${doubleValueChange[]} | Double | array of value changes for the given SQL query |
| \${intLastValue} | Integer | last integer value |
| \${doubleLastValue} | Double | last value |
| \${intLastValueChange} | Integer | last integer value change |
| \${doubleLastValueChange} | Double | last value change |

You can also display a single graph in your JSP files by including the file /trend/single-trend-box.jsp and specifying the name parameter.

Sample JSP snippet to include a single Trend chart with name 'example'

Chapter 4. Service Assurance

In *OpenNMS* the daemon to measures service availability and latency is done by *Pollerd*. To run these tests *Service Monitors* are scheduled and run in parallel in a *Thread Pool*. The behavior of *Pollerd* uses the following files for configuration and logging. Functionalities and general concepts are described in the *User Documentation* of *OpenNMS*. This section describes how to configure *Pollerd* for service assurance with all available *Service Monitors* coming with *OpenNMS*.

4.1. Pollerd Configuration

Table 8. Configuration and log files related to Pollerd.

| File | Description |
|--|--|
| <pre>\$OPENNMS_HOME/etc/poller- configuration.xml</pre> | Configuration file for monitors and global daemon configuration |
| <pre>\$OPENNMS_HOME/logs/poller.log</pre> | Log file for all monitors and the global <i>Pollerd</i> |
| <pre>\$OPENNMS_HOME/etc/response- graph.properties</pre> | RRD graph definitions for service response time measurements |
| <pre>\$OPENNMS_HOME/etc/events/opennms.e vents.xml</pre> | Event definitions for <i>Pollerd</i> , i.e. <i>nodeLostService</i> , interfaceDown or nodeDown |

To change the behavior for service monitoring, the poller-configuration.xml can be modified. The configuration file is structured in the following parts:

- *Global daemon config*: Define the size of the used *Thread Pool* to run *Service Monitors* in parallel. Define and configure the *Critical Service* for *Node Event Correlation*.
- Polling packages: Package to allow grouping of configuration parameters for Service Monitors.
- *Downtime Model*: Configure the behavior of *Pollerd* to run tests in case of an *Outage* is detected.
- *Monitor service association*: Based on the name of the service, the implementation for application or network management protocols are assigned.

Global configuration parameters for Pollerd

- ① Size of the *Thread Pool* to run *Service Monitors* in parallel
- 2 Enable or Disable Path Outage functionality based on a Critical Node in a network path
- ③ In case of unresponsive service services a *serviceUnresponsive* event is generated and not an outage. It prevents to apply the *Downtime Model* to retest the service after 30 seconds and prevents false alarms.

Configuration changes are applied by restarting *OpenNMS* and *Pollerd*. It is also possible to send an *Event* to *Pollerd* reloading the configuration. An *Event* can be sent on the *CLI* or the *Web User*

Interface.

Send configuration reload event on CLI

cd \$OPENNMS_HOME/bin
./send-event.pl uei.opennms.org/internal/reloadDaemonConfig --parm 'daemonName
Pollerd'

| Home / Admin / | Send Event | |
|---------------------------|---|----|
| Send Event to Op | penNMS | |
| Event | OpenNMS-defined internal event: reload specified daemon configuration | \$ |
| UUID | | |
| Node ID: | | |
| Source Hostname: | vagrant-ubuntu-trusty-64 | |
| Interface: | | |
| Service: | | |
| Parameters: | Name: daemonName Value: | |
| Description: | Add additional parameter | |
| | | |
| Description: | Select One | \$ |
| Operator Instructions: | | |
| | Reset | |
| Send Event »» | | |

Figure 10. Send configuration reload event with the Web User Interface

If you define **new** services in **poller-configuration.xml** a service restart of *OpenNMS* is necessary.

4.2. Critical Service

The *Critical Service* is used to correlate outages from *Services* to a *nodeDown* or *interfaceDown* event. It is a global configuration of *Pollerd* defined in *poller-configuration.xml*. The *OpenNMS* default configuration enables this behavior.

Critical Service Configuration in Pollerd

- ① Enable Node Outage correlation based on a Critical Service
- ② Optional: In case of nodes without a *Critical Service* this option controls the behavior. If set to true then all services will be polled. If set to false then the first service in the package that exists on the node will be polled until service is restored, and then polling will resume for all services.
- 3 Define Critical Service for Node Outage correlation

4.3. Downtime Model

By default the monitoring interval for a service is 5 minutes. To detect also short services outages, caused for example by automatic network rerouting, the downtime model can be used. On a detected service outage, the interval is reduced to 30 seconds for 5 minutes. If the service comes back within 5 minutes, a shorter outage is documented and the impact on service availability can be less than 5 minutes. This behavior is called *Downtime Model* and is configurable.



Figure 11. Downtime model with resolved and ongoing outage

In figure Outages and Downtime Model there are two outages. The first outage shows a short outage which was detected as *up* after 90 seconds. The second outage is not resolved now and the monitor has not detected an available service and was not available in the first 5 minutes (10 times 30 second polling). The scheduler changed the polling interval back to 5 minutes.

Example default configuration of the Downtime Model

```
<downtime interval="30000" begin="0" end="300000" /> ①
<downtime interval="300000" begin="300000" end="43200000" /> ②
<downtime interval="600000" begin="43200000" end="432000000" /> ③
<downtime begin="432000000" delete="true" /> ④
```

- ① from 0 seconds after an outage is detected until 5 minutes the polling interval will be set to 30 seconds
- ② after 5 minutes of an ongoing outage until 12 hours the polling interval will be set to 5 minutes
- 3 after 12 hours of an ongoing outage until 5 days the polling interval will be set to 10 minutes
- 4 after 5 days of an ongoing outage the service will be deleted from the monitoring system

4.4. Path Outages

To reduce the amount of alarms and notifications a *Path Outage* can be configured. This functionality is used to suppress *Notifications* based on the node depending on each other in the network path. The dependency is modeled in the *Node Provisioning* in *Path Outage*.



By default the *Path Outage* feature is disabled and has to be enabled in the pollerd-configuration.xml.

It requires the following information:

- Parent Foreign Source: The Foreign Source where the parent node is defined.
- Parent Foreign ID: The Foreign ID of the parent Node where this node depends on.
- The IP Interface selected as Primary is used as Critical IP

Additionally it is possible to define generic rules for *Path Outages*. For example there is a whole *IP Subnet* behind a *Router* and this *Router* is the *Critical Path* to this *IP Subnet*.

The configuration can be made in $Admin \rightarrow Configure \ Notifications \rightarrow Configure \ Path \ Outages$. It requires to specify a $Critical\ IP$ of the Router and allows to specify the IP Subnet by defining a Rule/Filter. They are specified in Rules/Filters in the $OpenNMS\ Wiki$. In this case, the Router with all Nodes on the $IP\ Subnet$ are down, but only one Notification is sent. All other $Node\ Down$ notifications are suppressed matching the Rule/Filter defined in the $Path\ Outage$.

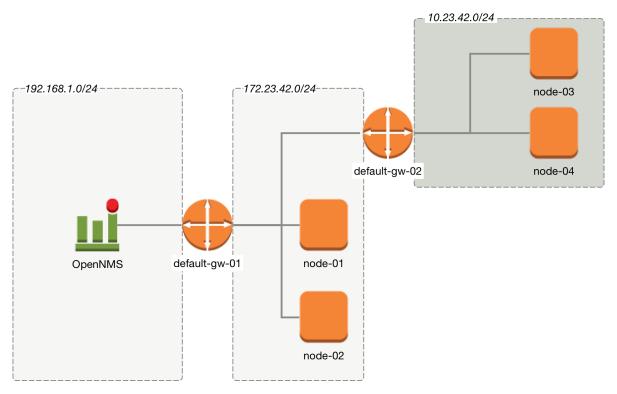


Figure 12. Topology for Path Outage

To configure a *Path Outage* based on the example in figure Topology for Path Outage, the configuration has to be defined as the following.



This example expects all *Nodes* are defined in the same *Foreign Source* named Network-ACME and the *Foreign ID* is the same as the *Node Label*.

Table 9. Provisioning for Topology Example

| Parent Foreign Source | Parent Foreign ID | Provisioned Node |
|--------------------------|----------------------|---------------------|
| not defined | not defined | default-gw-01 |
| Network-ACME | default-gw-01 | node-01 |
| Network-ACME | default-gw-01 | node-02 |
| Network-ACME | default-gw-01 | default-gw02 |
| Network-ACME | default-gw-02 | node-03 |
| Network-ACME | default-gw-02 | node-04 |



The *IP Interface* which is set to *Primary* is selected as the *Critical IP*. In this example it is important the *IP interface* on *default-gw-01* in the network 192.168.1.0/24 is set as *Primary* interface. The IP interface in the network 172.23.42.0/24 on *default-gw-02* is set as *Primary* interface.

4.5. Poller Packages

To define more complex monitoring configuration it is possible to group Service configurations into

Polling Packages. They allow to define assign to *Nodes* different *Service Configurations*. To assign a *Polling Package* to nodes the Rules/Filters syntax can be used. Each *Polling Package* can have its own Downtime Model configuration.

Multiple packages can be configured, and an interface can exist in more than one package. This gives great flexibility to how the service levels will be determined for a given device.

Polling package assigned to Nodes with Rules and Filters

- 1 Unique name of the polling package.
- ② Filter can be based on IP address, categories or asset attributes of *Nodes* based on Rules/Filters. The filter is evaluated first and is **required**. This package is used for all *IP Interfaces* which don't have 0.0.0.0 as an assigned *IP address* and is **required**.
- 3 Allow to specify if the configuration of *Services* is applied on a range of *IP Interfaces* (IPv4 or IPv6).

Instead of the include-range it is possible to add one or more specific *IP-Interfaces* with:

Defining a specific IP Interfaces

```
<specific>192.168.1.59</specific>
```

It is also possible to exclude *IP Interfaces* with:

Exclude IP Interfaces

```
<exclude-range begin="192.168.0.100" end="192.168.0.104"/>
```

4.5.1. Response Time Configuration

The definition of *Polling Packages* allows to configure similar services with different polling intervals. All the response time measurements are persisted in *RRD Files* and require a definition. Each *Polling Package* contains a *RRD* definition

- ① Polling interval for all services in this *Polling Package* is reflected in the step of size 300 seconds. All services in this package have to polled in 5 min interval, otherwise response time measurements are not correct persisted.
- 2 1 step size is persisted 2016 times: 1 * 5 min * 2016 = 7 d, 5 min accuracy for 7 d.
- 3 12 steps average persisted 1488 times: 12 * 5 min * 1488 = 62 d, aggregated to 60 min for 62 d.
- 4 288 steps average persisted 366 times: 288 * 5 min * 366 = 366 d, aggregated to 24 h for 366 d.
- ⑤ 288 steps maximum from 24 h persisted for 366 d.
- 6 288 steps minimum from 24 h persisted for 366 d.



The *RRD* configuration and the service polling interval has to be aligned. In other cases the persisted response time data is not correct displayed in the response time graph.



If the polling interval is changed afterwards, existing *RRD* files needs to be recreated with the new definitions.

4.5.2. Overlapping Services

With the possibility of specifying multiple *Polling Packages* it is possible to use the same *Service* like *ICMP* multiple times. The order how *Polling Packages* in the poller-configuration.xml are defined is important when *IP Interfaces* match multiple *Polling Packages* with the same *Service* configuration.

The following example shows which configuration is applied for a specific service:

```
<package name="less-specific">
 <filter>IPADDR != '0.0.0.0'</filter>
 <include-range begin="1.1.1.1" end="254.254.254.254" />
 <include-range begin="::1" end="ffff:ffff:ffff:ffff:ffff:ffff:ffff:/>
 <rrd step="300"> ①
   <rra>RRA:AVERAGE:0.5:1:2016
   <rra>RRA:AVFRAGF:0.5:12:1488
   <rra>RRA:AVERAGE:0.5:288:366</rra>
   <rra>RRA:MAX:0.5:288:366</rra>
   <rra>RRA:MIN:0.5:288:366</rra>
 <service name="ICMP" interval="300000" user-defined="false" status="on"> ②
   <parameter key="retry" value="5" /> ③
   <parameter key="timeout" value="10000" /> 4
   <parameter key="rrd-repository" value="/var/lib/opennms/rrd/response" />
   <parameter key="rrd-base-name" value="icmp" />
   <parameter key="ds-name" value="icmp" />
 </service>
 <downtime interval="30000" begin="0" end="300000" />
 <downtime interval="300000" begin="300000" end="43200000" />
 <downtime interval="600000" begin="43200000" end="432000000" />
</package>
<package name="more-specific">
 <filter>IPADDR != '0.0.0.0'</filter>
 <include-range begin="192.168.1.1" end="192.168.1.254" />
 <include-range begin="2600::1" end="2600:::fffff" />
 <rrd step="30"> ①
   <rra>RRA:AVERAGE:0.5:1:20160
   <rra>RRA:AVERAGE:0.5:12:14880</rra>
   <rra>RRA:AVERAGE:0.5:288:3660</rra>
   <rra>RRA:MAX:0.5:288:3660</rra>
   <rra>RRA:MIN:0.5:288:3660</rra>
 </rrd>
 <service name="ICMP" interval="30000" user-defined="false" status="on"> ②
   <parameter key="retry" value="2" /> ③
   <parameter key="timeout" value="3000" /> ④
   <parameter key="rrd-repository" value="/var/lib/opennms/rrd/response" />
   <parameter key="rrd-base-name" value="icmp" />
   <parameter key="ds-name" value="icmp" />
 </service>
 <downtime interval="10000" begin="0" end="300000" />
 <downtime interval="300000" begin="300000" end="43200000" />
 <downtime interval="600000" begin="43200000" end="43200000" />
</package>
```

- ① Polling interval in the packages are 300 seconds and 30 seconds
- ② Different polling interval for the service *ICMP*

- 3 Different retry settings for the service *ICMP*
- 4 Different timeout settings for the service *ICMP*

The last *Polling Package* on the service will be applied. This can be used to define a less specific catch all filter for a default configuration. A more specific *Polling Package* can be used to overwrite the default setting. In the example above all *IP Interfaces* in 192.168.1/24 or 2600:/64 will be monitored with ICMP with different polling, retry and timeout settings.

Which *Polling Packages* are applied to the *IP Interface* and *Service* can be found in the *Web User Interface*. The *IP Interface* and *Service* page show which *Polling Package* and *Service* configuration is applied for this specific service.

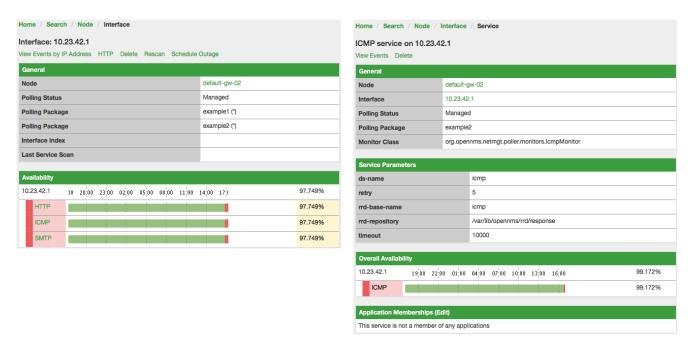


Figure 13. Polling Package applied to IP interface and Service

4.5.3. Test Services on manually

For troubleshooting it is possible to run a test via the *Karaf Shell*:

ssh -p 8101 admin@localhost

Once in the shell, you can print show the commands help as follows:

```
opennms> poller:test --help
DESCRIPTION
        poller:test
        Execute a poller test from the command line using current settings from
poller-configuration.xml
SYNTAX
        poller:test [options]
OPTIONS
        -s, --service
                Service name
        -p, --param
                Service parameter ~ key=value
        -i, --ipaddress
                IP Address to test
        -P, --package
                Poller Package
        -c, --class
                Monitor Class
        --help
                Display this help message
```

The following example runs the *ICMP* monitor on a specific *IP Interface*.

Run ICMP monitor configuration defined in specific Polling Package

```
opennms> poller:test -i 10.23.42.1 -s ICMP -P example1
```

The output is verbose which allows debugging of *Monitor* configurations. Important output lines are shown as the following:

Important output testing a service on the CLI

```
Checking service ICMP on IP 10.23.42.1 ①
Package: example1 ②
Monitor: org.opennms.netmgt.poller.monitors.IcmpMonitor ③
Parameter ds-name : icmp ④
Parameter rrd-base-name : icmp ④
Parameter rrd-repository : /var/lib/opennms/rrd/response ④
Parameter retry : 2 ⑤
Parameter timeout : 3000 ⑤

Available ? true (status Up[1])
```

- ① Service and IP Interface to run the test
- ② Applied Service configuration from Polling Package for this test

- 3 Service Monitor used for this test
- 4 RRD configuration for response time measurement
- (5) Retry and timeout settings for this test

4.6. Service monitors

To support several specific applications and management agents, *Pollerd* executes *Service Monitors*. This section describes all available built-in *Service Monitors* which are available and can be configured to allow complex monitoring. For information how these can be extended, see *Development Guide* of the *OpenNMS* documentation.

4.6.1. AvailabilityMonitor

This monitor tests reachability of a node by using the *isReachable* method of the *InetAddress* java class. The service is considered available if isReachable returns true. See Oracle's documentation for more details.



This monitor is deprecated in favour of the IcmpMonitor monitor. You should only use this monitor on remote pollers running on unusual configurations (See below for more details).

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.AvailabilityMonitor |
|----------------|--|
| Remote Enabled | true |

Configuration and Usage

Table 10. Monitor specific parameters for the AvailabilityMonitor

| Paramete r | Description | Require d | Default value |
|---------------|---|--------------|------------------|
| retry | Number of attempts to have the <i>isReachable</i> method return <i>true</i> . | optional | 3 |
| timeout | Timeout for the isReachable method, in milliseconds. | optional | 3000 |

Examples

```
<service name="AVAIL" interval="300000" user-defined="false" status="on">
   <parameter key="retry" value="2"/>
        <parameter key="timeout" value="5000"/>
   </service>

<monitor service="AVAIL" class-name=
   "org.opennms.netmgt.poller.monitors.AvailabilityMonitor"/>
```

IcmpMonitor vs AvailabilityMonitor

This monitor has been developped in a time when the IcmpMonitor monitor wasn't remote enabled, to circumvent this limitation. Now, with the JNA ICMP implementation, the IcmpMonitor monitor is remote enabled under most configurations and this monitor shouldn't be needed -unless you're running your remote poller on such an unusual configuration (See also issue NMS-6735 for more information)-.

4.6.2. BgpSessionMonitor

This monitor checks if a BGP-Session to a peering partner (peer-ip) is functional. To monitor the BGP-Session the RFC1269 SNMP MIB is used and test the status of the session using the following OIDs is used:

```
BGP_PEER_STATE_OID = .1.3.6.1.2.1.15.3.1.2.<peer-ip>
BGP_PEER_ADMIN_STATE_OID = .1.3.6.1.2.1.15.3.1.3.<peer-ip>
BGP_PEER_REMOTEAS_OID = .1.3.6.1.2.1.15.3.1.9.<peer-ip>
BGP_PEER_LAST_ERROR_OID = .1.3.6.1.2.1.15.3.1.14.<peer-ip>
BGP_PEER_FSM_EST_TIME_OID = .1.3.6.1.2.1.15.3.1.16.<peer-ip>
```

The <peer-ip> is the far end IP address of the BGP session end point.

A SNMP get request for BGP_PEER_STATE_OID returns a result between 1 to 6. The servicestates for OpenNMS Meridian are mapped as follows:

| Resul t | State description | Monitor state in OpenNMS Meridian |
|------------|----------------------|--------------------------------------|
| 1 | Idle | DOWN |
| 2 | Connect | DOWN |
| 3 | Active | DOWN |
| 4 | OpenSent | DOWN |
| 5 | OpenConfirm | DOWN |
| 6 | Established | UP |

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.BgpSessionMonitor |
|----------------|--|
| Remote Enabled | false |

To define the mapping I used the description from RFC1771 BGP Finite State Machine.

Configuration and Usage

| Paramet er | Description | Require d | Default value |
|---------------|---|--------------|------------------|
| bgpPeerIp | IP address of the far end BGP peer session | require d | - |
| retry | Amount of attempts to get the BGP peer state with SNMP | require d | - |
| timeout | Time to wait for the SNMP agents response before trying a next attempt. | require d | - |

Examples

To monitor the session state *Established* it is necessary to add a service to your poller configuration in '\$OPENNMS_HOME/etc/poller-configuration.xml', for example:

Error code mapping

The *BGP_PEER_LAST_ERROR_OID* gives an error in HEX-code. To make it human readable a codemapping table is implemented:

| Error code | Error Message |
|---------------|---|
| 0100 | Message Header Error |
| 0101 | Message Header Error - Connection Not Synchronized |
| 0102 | Message Header Error - Bad Message Length |
| 0103 | Message Header Error - Bad Message Type |
| 0200 | OPEN Message Error |
| 0201 | OPEN Message Error - Unsupported Version Number |
| 0202 | OPEN Message Error - Bad Peer AS |
| 0203 | OPEN Message Error - Bad BGP Identifier |
| 0204 | OPEN Message Error - Unsupported Optional Parameter |

| Error code | Error Message |
|------------|--|
| 0205 | OPEN Message Error (deprecated) |
| 0206 | OPEN Message Error - Unacceptable Hold Time |
| 0300 | UPDATE Message Error |
| 0301 | UPDATE Message Error - Malformed Attribute List |
| 0302 | UPDATE Message Error - Unrecognized Well-known Attribute |
| 0303 | UPDATE Message Error - Missing Well-known Attribute |
| 0304 | UPDATE Message Error - Attribute Flags Error |
| 0305 | UPDATE Message Error - Attribute Length Error |
| 0306 | UPDATE Message Error - Invalid ORIGIN Attribute |
| 0307 | UPDATE Message Error (deprecated) |
| 0308 | UPDATE Message Error - Invalid NEXT_HOP Attribute |
| 0309 | UPDATE Message Error - Optional Attribute Error |
| 030A | UPDATE Message Error - Invalid Network Field |
| 030B | UPDATE Message Error - Malformed AS_PATH |
| 0400 | Hold Timer Expired |
| 0500 | Finite State Machine Error |
| 0600 | Cease |
| 0601 | Cease - Maximum Number of Prefixes Reached |
| 0602 | Cease - Administrative Shutdown |
| 0603 | Cease - Peer De-configured |
| 0604 | Cease - Administrative Reset |
| 0605 | Cease - Connection Rejected |
| 0606 | Cease - Other Configuration Change |
| 0607 | Cease - Connection Collision Resolution |
| 0608 | Cease - Out of Resources |

Instead of HEX-Code the error message will be displayed in the service down logmessage. To give some additional informations the logmessage contains also

```
BGP-Peer Adminstate
BGP-Peer Remote AS
BGP-Peer established time in seconds
```

Debugging

If you have problems to detect or monitor the BGP Session you can use the following command to figure out where the problem come from.

```
snmpwalk -v 2c -c <myCommunity> <myRouter2Monitor> .1.3.6.1.2.1.15.3.1.2.99.99.99.99
```

Replace 99.99.99 with your BGP-Peer IP. The result should be an Integer between 1 and 6.

4.6.3. BSFMonitor

This monitor runs a *Bean Scripting Framework* BSF compatible script to determine the status of a service. Users can write scripts to perform highly custom service checks. This monitor is not optimised for scale. It's intended for a small number of custom checks or prototyping of monitors.

BSFMonitor vs SystemExecuteMonitor

The *BSFMonitor* avoids the overhead of *fork(2)* that is used by the *SystemExecuteMonitor*. *BSFMonitor* also grants access to a selection of *OpenNMS Meridian* internal methods and classes that can be used in the script.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.BSFMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

Table 11. Monitor specific parameters for the BSFMonitor

| Paramet er | Description | Requ ired | Default value |
|-------------------------|---|--------------|---|
| file-name | Path to the script file. | requi red | - |
| bsf- engine | The BSF Engine to run the script in different languages like Bean Shell: bsh.util.BeanShellBSFEngine Groovy: org.codehaus.groovy.bsf.GroovyEngine Jython: org.apache.bsf.engines.jython.JythonEngine | requi red | - |
| run-type | one of eval or exec | optio nal | eval |
| lang- class | The BSF language class, like groovy or beanshell. | optio nal | file-name extension is interpreted by default |
| file- extension s | comma-separated list | optio nal | - |

Table 12. Beans which can be used in the script

| Variabl e | Туре | Description |
|-----------------|--|--|
| map | Map <string, object=""></string,> | The <i>map</i> contains all various parameters passed to the monitor from the service definition it the poller-configuration.xml file. |
| ip_addr | String | The IP address that is currently being polled. |
| node_id | int | The Node ID of the node the ip_addr belongs to. |
| node_la bel | String | The Node Label of the node the <code>ip_addr</code> and service belongs to. |
| svc_nam e | String | The name of the service that is being polled. |
| bsf_mon itor | BSFMonitor | The instance of the <i>BSFMonitor</i> object calling the script. Useful for logging via its log(String sev, String fmt, Object args) method. |
| results | HashMap <string, String></string, | The script is expected to put its results into this object. The status indication should be set into the entry with key status. If the status is not OK, a key reason should contain a description of the problem. |
| times | LinkedHashMap <strin g, Number></strin | The script is expected to put one or more response times into this object. |

Additionally every parameter added to the service definition in poller-configuration.xml is available as a *String* object in the script. The key attribute of the parameter represents the name of the *String* object and the value attribute represents the value of the *String* object.



Please keep in mind, that these parameters are also accessible via the *map* bean.



Avoid non-character names for parameters to avoid problems in the script languages.

Response Codes

The script has to provide a status code that represents the status of the associated service. The following status codes are defined:

Table 13. Status codes

| Cod e | Description |
|----------|-------------------------|
| OK | Service is available |
| UNK | Service status unknown |
| UNR | Service is unresponsive |
| NOK | Service is unavailable |

Response time tracking

By default the *BSFMonitor* tracks the whole time the script file consumes as the response time. If the response time should be persisted the response time add the following parameters:

RRD response time tracking for this service in poller-configuration.xml

```
<!-- where in the filesystem response times are stored -->
<parameter key="rrd-repository" value="/opt/opennms/share/rrd/response" />

<!-- name of the rrd file -->
<parameter key="rrd-base-name" value="minimalbshbase" />

<!-- name of the data source in the rrd file -->
<!-- by default "response-time" is used as ds-name -->
<parameter key="ds-name" value="myResponseTime" />
```

It is also possible to return one or many response times directly from the script. To add custom response times or override the default one, add entries to the *times* object. The entries are keyed with a *String* that names the datasource and have as values a number that represents the response time. To override the default response time datasource add an entry into times named response-time.

Timeout and Retry

The *BSFMonitor* does not perform any timeout or retry processing on its own. If retry and or timeout behaviour is required, it has to be implemented in the script itself.

Requirements for the script (run-types)

Depending on the run-type the script has to provide its results in different ways. For minimal scripts with very simple logic run-type eval is the simple option. Scripts running in eval mode have to return a *String* matching one of the status codes.

If your script is more than a one-liner, run-type exec is essentially required. Scripts running in exec mode need not return anything, but they have to add a status entry with a status code to the results object. Additionally, the results object can also carry a "reason": "message" entry that is used in non OK states.

Commonly used language settings

The *BSF* supports many languages, the following table provides the required setup for commonly used languages.

Table 14. BSF language setups

| Languag e | lang- class | bsf-engine | required library |
|--------------|----------------|-----------------------------|----------------------|
| BeanShell | beanshell | bsh.util.BeanShellBSFEngine | supported by default |

| Languag e | lang- class | bsf-engine | required library |
|--------------|----------------|--|--------------------------|
| Groovy | groovy | org.codehaus.groovy.bsf.GroovyEngine | groovy-all-[version].jar |
| Jython | jython | org.apache.bsf.engines.jython.JythonEngine | jython-[version].jar |

Example Bean Shell

BeanShell example poller-configuration.xml

```
<service name="MinimalBeanShell" interval="300000" user-defined="true" status="on">
    <parameter key="file-name" value="/tmp/MinimalBeanShell.bsh"/>
    <parameter key="bsf-engine" value="bsh.util.BeanShellBSFEngine"/>
    </service>

<monitor service="MinimalBeanShell" class-name=
    "org.opennms.netmgt.poller.monitors.BSFMonitor" />
```

BeanShell example MinimalBeanShell.bsh script file

```
bsf_monitor.log("ERROR", "Starting MinimalBeanShell.bsf", null);
File testFile = new File("/tmp/TestFile");
if (testFile.exists()) {
   return "OK";
} else {
   results.put("reason", "file does not exist");
   return "NOK";
}
```

Example Groovy

To use the Groovy language an additional library is required. Copy a compatible groovy-all.jar into to opennms/lib folder and restart *OpenNMS Meridian*. That makes *Groovy* available for the *BSFMonitor*.

Groovy example poller-configuration.xml with default run-type set to eval

```
<service name="MinimalGroovy" interval="300000" user-defined="true" status="on">
    <parameter key="file-name" value="/tmp/MinimalGroovy.groovy"/>
    <parameter key="bsf-engine" value="org.codehaus.groovy.bsf.GroovyEngine"/>
    </service>

<monitor service="MinimalGroovy" class-name=
    "org.opennms.netmgt.poller.monitors.BSFMonitor" />
```

Groovy example MinimalGroovy.groovy script file for run-type eval

```
bsf_monitor.log("ERROR", "Starting MinimalGroovy.groovy", null);
File testFile = new File("/tmp/TestFile");
if (testFile.exists()) {
   return "OK";
} else {
   results.put("reason", "file does not exist");
   return "NOK";
}
```

Groovy example poller-configuration.xml with run-type set to exec

```
<service name="MinimalGroovy" interval="300000" user-defined="true" status="on">
    <parameter key="file-name" value="/tmp/MinimalGroovy.groovy"/>
    <parameter key="bsf-engine" value="org.codehaus.groovy.bsf.GroovyEngine"/>
    <parameter key="run-type" value="exec"/>
    </service>

<monitor service="MinimalGroovy" class-name=
    "org.opennms.netmgt.poller.monitors.BSFMonitor" />
```

Groovy example MinimalGroovy.groovy script file for run-type set to exec

```
bsf_monitor.log("ERROR", "Starting MinimalGroovy", null);
def testFile = new File("/tmp/TestFile");
if (testFile.exists()) {
   results.put("status", "OK")
} else {
   results.put("reason", "file does not exist");
   results.put("status", "NOK");
}
```

Example Jython

To use the *Jython* (*Java* implementation of *Python*) language an additional library is required. Copy a compatible jython-x.y.z.jar into the opennms/lib folder and restart *OpenNMS Meridian*. That makes *Jython* available for the *BSFMonitor*.

Tython example poller-configuration.xml *with* run-type exec

```
<service name="MinimalJython" interval="300000" user-defined="true" status="on">
    <parameter key="file-name" value="/tmp/MinimalJython.py"/>
    <parameter key="bsf-engine" value="org.apache.bsf.engines.jython.JythonEngine"/>
    <parameter key="run-type" value="exec"/>
    </service>

<monitor service="MinimalJython" class-name=
    "org.opennms.netmgt.poller.monitors.BSFMonitor" />
```

Jython example MinimalJython.py script file for run-type set to exec

```
from java.io import File

bsf_monitor.log("ERROR", "Starting MinimalJython.py", None);
if (File("/tmp/TestFile").exists()):
        results.put("status", "OK")
else:
    results.put("reason", "file does not exist")
    results.put("status", "NOK")
```

- We have to use run-type exec here because *Jython* chokes on the import keyword in eval mode.
- As profit that this is really *Python*, notice the substitution of *Python's* None value for Java's null in the log call.

Advanced examples

The following example references all beans that are exposed to the script, including a custom parameter.

```
<service name="MinimalGroovy" interval="30000" user-defined="true" status="on">
    <parameter key="file-name" value="/tmp/MinimalGroovy.groovy"/>
    <parameter key="bsf-engine" value="org.codehaus.groovy.bsf.GroovyEngine"/>

<!-- custom parameters (passed to the script) -->
    <parameter key="myParameter" value="Hello Groovy" />

<!-- optional for response time tracking -->
    <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response" />
    <parameter key="rrd-base-name" value="minimalgroovybase" />
    <parameter key="ds-name" value="minimalgroovyds" />
    </service>

<monitor service="MinimalGroovy" class-name=
    "org.opennms.netmgt.poller.monitors.BSFMonitor" />
```

Groovy example Bean referencing script file

```
bsf_monitor.log("ERROR", "Starting MinimalGroovy", null);
//list of all available objects from the BSFMonitor
Map<String, Object> map = map;
bsf_monitor.log("ERROR", "---- map ----", null);
bsf monitor.log("ERROR", map.toString(), null);
String ip_addr = ip_addr;
bsf_monitor.log("ERROR", "---- ip_addr ----", null);
bsf_monitor.log("ERROR", ip_addr, null);
int node id = node id;
bsf_monitor.log("ERROR", "---- node_id ----", null);
bsf_monitor.log("ERROR", node_id.toString(), null);
String node_label = node_label;
bsf monitor.log("ERROR", "---- node label ----", null);
bsf monitor.log("ERROR", node label, null);
String svc name = svc name;
bsf_monitor.log("ERROR", "---- svc_name ----", null);
bsf_monitor.log("ERROR", svc_name, null);
org.opennms.netmgt.poller.monitors.BSFMonitor bsf_monitor = bsf_monitor;
bsf_monitor.log("ERROR", "---- bsf_monitor ----", null);
bsf_monitor.log("ERROR", bsf_monitor.toString(), null);
HashMap<String, String> results = results;
bsf_monitor.log("ERROR", "---- results ----", null);
bsf_monitor.log("ERROR", results.toString(), null);
```

```
LinkedHashMap<String, Number> times = times;
bsf_monitor.log("ERROR", "---- times ----", null);
bsf_monitor.log("ERROR", times.toString(), null);
// reading a parameter from the service definition
String myParameter = myParameter;
bsf_monitor.log("ERROR", "---- myParameter ----", null);
bsf monitor.log("ERROR", myParameter, null);
// minimal example
def testFile = new File("/tmp/TestFile");
if (testFile.exists()) {
  bsf_monitor.log("ERROR", "Done MinimalGroovy ---- OK ----", null);
  return "OK";
} else {
  results.put("reason", "file does not exist");
  bsf_monitor.log("ERROR", "Done MinimalGroovy ---- NOK ----", null);
  return "NOK";
}
```

4.6.4. CiscoIpSlaMonitor

This monitor can be used to monitor IP SLA configurations on your Cisco devices. This monitor supports the following SNMP OIDS from CISCO-RTT-MON-MIB:

```
RTT_ADMIN_TAG_OID = .1.3.6.1.4.1.9.9.42.1.2.1.1.3

RTT_OPER_STATE_OID = .1.3.6.1.4.1.9.9.42.1.2.9.1.10

RTT_LATEST_OPERSENSE_OID = .1.3.6.1.4.1.9.9.42.1.2.10.1.2

RTT_ADMIN_THRESH_OID = .1.3.6.1.4.1.9.9.42.1.2.1.1.5

RTT_ADMIN_TYPE_OID = .1.3.6.1.4.1.9.9.42.1.2.1.1.4

RTT_LATEST_OID = .1.3.6.1.4.1.9.9.42.1.2.10.1.1
```

The monitor can be run in two scenarios. The first one tests the $RTT_LATEST_OPERSENSE$ which is a sense code for the completion status of the latest RTT operation. If the $RTT_LATEST_OPERSENSE$ returns ok(1) the service is marked as up.

The second scenario is to monitor the configured threshold in the *IP SLA* config. If the *RTT_LATEST_OPERSENSE* returns with *overThreshold(3)* the service is marked *down*.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.CiscoIpSlaMonitor |
|----------------|--|
| Remote Enabled | false |

Configuration and Usage

Table 15. Monitor-specific parameters for the CiscoIpSlaMonitor

| Paramet er | Description | Requi red | Default value |
|-------------------|--|--------------|--------------------------|
| retry | Number of retries to get the information from the SNMP agent before the service is marked as down. | option al | from snmp- config.xml |
| timeout | Time in milliseconds to wait for the result from the SNMP agent before making the next attempt. | option al | from snmp- config.xml |
| admin-tag | The tag attribute from your <i>IP SLA</i> configuration you want to monitor. | requir ed | - |
| ignore- thresh | Boolean indicates if just the status or configured threshold should be monitored. | requir ed | |

Example for HTTP and ICMP echo reply

In this example we configure an IP SLA entry to monitor Google's website with *HTTP GET* from the Cisco device. We use 8.8.8.8 as our DNS resolver. In our example our SLA says we should reach Google's website within 200ms. To advise co-workers that this monitor entry is used for monitoring, I set the owner to *OpenNMS*. The tag is used to identify the entry later in the SNMP table for monitoring.

Cisco device configuration for IP SLA instance for HTTP GET

```
ip sla monitor 1
type http operation get url http://www.google.de name-server 8.8.8.8
timeout 3000
threshold 200
owner OpenNMS
tag Google Website
ip sla monitor schedule 3 life forever start-time now
```

In the second example we configure a IP SLA to test if the IP address from www.opennms.org is reachable with ICMP from the perspective of the Cisco device. Like the example above we have a threshold and a timeout.

Cisco device configuration for IP SLA instance for ICMP monitoring.

```
ip sla 1
icmp-echo 64.146.64.212
timeout 3000
threshold 150
owner OpenNMS
tag OpenNMS Host
ip sla schedule 1 life forever start-time now
```



It's not possible to reconfigure an IP SLA entry. If you want to change parameters, you have to delete the whole configuration and reconfigure it with your new parameters. Backup your Cisco configuration manually or take a look at RANCID.

To monitor both of the entries the configuration in poller-configuration.xml requires two service definition entries:

```
<service name="IP-SLA-WEB-Google" interval="300000"</pre>
 user-defined="false" status="on">
    <parameter key="retry" value="2" />
    <parameter key="timeout" value="3000" />
    <parameter key="admin-tag" value="Google Website" />
    <parameter key="ignore-thresh" value="false" /> ①
</service>
<service name="IP-SLA-PING-OpenNMS" interval="300000"</pre>
 user-defined="false" status="on">
    <parameter key="retry" value="2" />
    <parameter key="timeout" value="3000" />
    <parameter key="admin-tag" value="OpenNMS Host" />
    <parameter key="ignore-thresh" value="true" /> ②
</service>
<monitor service="IP-SLA-WEB-Google" class-name=</pre>
"org.opennms.netmgt.poller.monitors.CiscoIpSlaMonitor" />
<monitor service="IP-SLA-PING-OpenNMS" class-name=</pre>
"org.opennms.netmgt.poller.monitors.CiscoIpSlaMonitor" />
```

- ① Service is *up* if the IP SLA state is *ok(1)*
- ② Service is *down* if the IP SLA state is *overThreshold(3)*

4.6.5. CiscoPingMibMonitor

This poller monitor's purpose is to create conceptual rows (entries) in the *ciscoPingTable* on *Cisco IOS* devices that support the CISCO-PING-MIB. These entries direct the remote *IOS* device to ping an IPv4 or IPv6 address with a configurable set of parameters. After the *IOS* device has completed the requested ping operations, the poller monitor queries the *IOS* device to determine the results. If the results indicate success according to the configured parameters in the service configuration, then the monitored service is reported as available and the results are available for optional time-series (RRD) storage. If the results indicate failure, the monitored service is reported unavailable with a descriptive reason code. If something goes wrong during the setup of the entry or the subsequent querying of its status, the monitored service is reported to be in an *unknown* state.



Unlike most poller monitors, the *CiscoPingMibMonitor* does not interpret the timeout and retries parameters to determine when a poll attempt has timed out or whether it should be attempted again. The packet-count and packet-timeout parameters instead service this purpose from the perspective of the remote *IOS* device.

Supported MIB OIDs from CISCO_PING_MIB

```
ciscoPingEntry
                           1.3.6.1.4.1.9.9.16.1.1.1
ciscoPingSerialNumber
                           1.3.6.1.4.1.9.9.16.1.1.1.1
ciscoPingProtocol
                           1.3.6.1.4.1.9.9.16.1.1.1.2
ciscoPingAddress
                           1.3.6.1.4.1.9.9.16.1.1.1.3
ciscoPingPacketCount
                           1.3.6.1.4.1.9.9.16.1.1.1.4
ciscoPingPacketSize
                           1.3.6.1.4.1.9.9.16.1.1.1.5
ciscoPingPacketTimeout
                           1.3.6.1.4.1.9.9.16.1.1.1.6
ciscoPingDelay
                           1.3.6.1.4.1.9.9.16.1.1.1.7
ciscoPingTrapOnCompletion
                           1.3.6.1.4.1.9.9.16.1.1.1.8
ciscoPingSentPackets
                           1.3.6.1.4.1.9.9.16.1.1.1.9
ciscoPingReceivedPackets
                           1.3.6.1.4.1.9.9.16.1.1.1.10
ciscoPingMinRtt
                           1.3.6.1.4.1.9.9.16.1.1.1.11
ciscoPingAvgRtt
                           1.3.6.1.4.1.9.9.16.1.1.1.12
ciscoPingMaxRtt
                           1.3.6.1.4.1.9.9.16.1.1.1.13
ciscoPingCompleted
                           1.3.6.1.4.1.9.9.16.1.1.1.14
ciscoPingEntryOwner
                           1.3.6.1.4.1.9.9.16.1.1.1.15
ciscoPingEntryStatus
                           1.3.6.1.4.1.9.9.16.1.1.1.16
ciscoPingVrfName
                           1.3.6.1.4.1.9.9.16.1.1.1.17
```

Prerequisites

- One or more *Cisco* devices running an *IOS* image of recent vintage; any 12.2 or later image is probably fine. Even very low-end devices appear to support the CISCO-PING-MIB.
- The *IOS* devices that will perform the remote pings must be configured with an *SNMP write* community string whose source address access-list includes the address of the OpenNMS Meridian server and whose MIB view (if any) includes the OID of the *ciscoPingTable*.
- The corresponding *SNMP write community* string must be specified in the write-community attribute of either the top-level <snmp-config> element of snmp-config.xml or a <definition> child element that applies to the *SNMP-primary* interface of the *IOS* device(s) that will perform the remote pings.

Scalability concerns

This monitor spends a fair amount of time sleeping while it waits for the remote *IOS* device to complete the requested ping operations. The monitor is pessimistic in calculating the delay between creation of the *ciscoPingTable* entry and its first attempt to retrieve the results of that entry's ping operations—it will always wait at least (packet-count * (packet-timeout + packet-delay)) milliseconds before even checking whether the remote pings have completed. It's therefore prone to hogging poller threads if used with large values for the packet-count, packet-timeout, and/or packet-delay parameters. Keep these values as small as practical to avoid tying up poller threads

unnecessarily.

This monitor always uses the current time in whole seconds since the UNIX epoch as the instance identifier of the *ciscoPingTable* entries that it creates. The object that holds this identifier is a signed 32-bit integer type, precluding a finer resolution. It's probably a good idea to mix in the least-significant byte of the millisecond-accurate time as a substitute for that of the whole-second-accurate value to avoid collisions. *IOS* seems to clean up entries in this table within a manner of minutes after their ping operations have completed.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.CiscoPingMibMonitor |
|----------------|--|
| Remote Enabled | false |

Configuration and Usage

Table 16. Monitor specific parameters for the CiscoPingMibMonitor

| Parameter | Description | Requ ired | Default value |
|----------------|--|--------------|------------------------------------|
| timeout | A timeout, in milliseconds, that should override the SNMP timeout specified in snmp-config.xml. Do not use without a very good reason to do so. | optio nal | from snmp- config.xml |
| retry | Number of retries to attempt if the initial attempt times out. Overrides the equivalent value from snmp-config.xml. Do not use unless really needed. | optio nal | from snmp- config.xml |
| version | SNMP protocol version (1, 2c, or 3) to use for operations performed by this service monitor. Do not use with out a very good reason to do so. | optio nal | from snmp- config.xml |
| packet-count | Number of ping packets that the remote <i>IOS</i> device should send. | optio nal | 5 |
| packet-size | Size, in bytes, of each ping packet that the remote <i>IOS</i> device should send. | optio nal | 100 |
| packet-timeout | Timeout, in milliseconds, of each ping packet sent by the remote <i>IOS</i> device. | optio nal | 2000 |
| packet-delay | Delay, in milliseconds, between ping packets sent by the remote <i>IOS</i> device. | optio nal | 0 |
| entry-owner | String value to set as the value of ciscoPingEntryOwner of entries created for this service. | optio nal | OpenNMS CiscoPingMibMo nitor |

| Parameter | Description | Requ ired | Default value |
|--|---|--------------|---------------|
| vrf-name | String value to set as the VRF (VLAN) name in whose context the remote <i>IOS</i> device should perform the pings for this service. | optio nal | empty String |
| proxy-node-id | Numeric database identifier of the node whose primary SNMP interface should be used as the <i>proxy</i> for this service. If specified along with the related proxy-node-foreign-source, proxy-node-foreign-id, and/or proxy-ip-addr, this parameter will be the effective one. | optio nal | |
| proxy-node-foreign- source proxy-node-foreign-id | foreign-source name and foreign-ID of the node whose primary SNMP interface should be used as the "proxy" for this service. These two parameters are corequisites. If they appear along with the related proxy-ip-addr, these parameters will be the effective ones. | optio nal | _ |
| proxy-ip-addr | IP address of the interface that should be used as the <i>proxy</i> for this service. Effective only if none of proxy-node-id, proxy-node-foreign-source, nor proxy-node-foreign-id appears alongside this parameter. A value of \${ipaddr} will be substituted with the IP address of the interface on which the monitored service appears. | optio nal | |
| target-ip-addr | IP address that the remote <i>IOS</i> device should ping. A value of \${ipaddr} will be substituted with the IP address of the interface on which the monitored service appears. | optio nal | - |
| success-percent | A whole-number percentage of pings that must succeed (from the perspective of the remote <i>IOS</i> device) in order for this service to be considered available. As an example, if packet-count is left at its default value of 5 but you wish the service to be considered available even if only one of those five pings is successful, then set this parameter's value to 20. | optio nal | 100 |
| rrd-repository | Base directory of an RRD repository in which to store this service monitor's response-time samples | optio nal | - |

| Parameter | Description | Requ ired | Default value |
|-----------|---|--------------|---------------|
| ds-name | Name of the RRD datasource (DS) name in which to store this service monitor's response-time samples; rrd-base-name Base name of the RRD file (minus the .rrd or .jrb file extension) within the specified rrd-repository path in which this service monitor's response-time samples will be persisted | optio nal | |

This is optional just if you can use variables in the configuration

Table 17. Variables which can be used in the configuration

| Variabl e | Description | |
|----------------|--|--|
| \${ipaddr } | This value will be substituted monitored service | d with the IP address of the interface on which the appears. |

Example: Ping the same non-routable address from all routers of customer Foo

A service provider's client, Foo Corporation, has network service at multiple locations. At each Foo location, a point-of-sale system is statically configured at IPv4 address 192.168.255.1. Foo wants to be notified any time a point-of-sale system becomes unreachable. Using an OpenNMS Meridian remote location monitor is not feasible. All of Foo Corporation's CPE routers must be *Cisco IOS* devices in order to achieve full coverage in this scenario.

One approach to this requirement is to configure all of Foo Corporation's premise routers to be in the surveillance categories Customer_Foo, CPE, and Routers, and to use a filter to create a poller package that applies only to those routers. We will use the special value ${ipaddr}$ for the proxy-ipaddr parameter so that the remote pings will be provisioned on each Foo CPE router. Since we want each Foo CPE router to ping the same IP address 192.168.255.1, we statically list that value for the target-ip-addr address.

```
<package name="ciscoping-foo-pos">
 <filter>catincCustomer_Foo & catincCPE & catincRouters & nodeSysOID LIKE
'.1.3.6.1.4.1.9.%'</filter>
 <include-range begin="0.0.0.0" end="254.254.254.254" />
 <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:MAX:0.5:288:366</rra>
    <rra>RRA:MIN:0.5:288:366</rra>
 <service name="FooPOS" interval="300000" user-defined="false" status="on">
    <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response" />
    <parameter key="rrd-base-name" value="ciscoping" />
    <parameter key="ds-name" value="ciscoping" />
    <parameter key="proxy-ip-addr" value="${ipaddr}" />
    <parameter key="target-ip-addr" value="192.168.255.1" />
 </service>
 <downtime interval="30000" begin="0" end="300000" /><!-- 30s, 0, 5m -->
 <downtime interval="300000" begin="300000" end="43200000" /><!-- 5m, 5m, 12h -->
 <downtime interval="600000" begin="43200000" end="432000000" /><!-- 10m, 12h, 5d -->
 <downtime begin="432000000" delete="true" /><!-- anything after 5 days delete -->
</package>
<monitor service="FooPOS" class-name=</pre>
"org.opennms.netmgt.poller.monitors.CiscoPingMibMonitor" />
```

Example: Ping from a single IOS device routable address of each router of customer Bar

A service provider's client, Bar Limited, has network service at multiple locations. While OpenNMS Meridian' world-class service assurance is generally sufficient, Bar also wants to be notified any time a premise router at one of their locations unreachable from the perspective of an *IOS* device in Bar's main data center. Some or all of the Bar Limited CPE routers may be non-Cisco devices in this scenario.

To meet this requirement, our approach is to configure Bar Limited's premise routers to be in the surveillance categories Customer_Bar, CPE, and Routers, and to use a filter to create a poller package that applies only to those routers. This time, though, we will use the special value \${ipaddr} not in the proxy-ip-addr parameter but in the target-ip-addr parameter so that the remote pings will be performed for each Bar CPE router. Since we want the same *IOS* device 20.11.5.11 to ping the CPE routers, we statically list that value for the proxy-ip-addr address. Example pollerconfiguration.xml additions

```
<package name="ciscoping-bar-cpe">
 <filter>catincCustomer Bar & catincCPE & catincRouters</filter>
 <include-range begin="0.0.0.0" end="254.254.254.254" />
 <rrd step="300">
   <rra>RRA:AVERAGE:0.5:1:2016
   <rra>RRA:AVERAGE:0.5:12:1488
   <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>
 <service name="BarCentral" interval="300000" user-defined="false" status="on">
   <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response" />
   <parameter key="rrd-base-name" value="ciscoping" />
   <parameter key="ds-name" value="ciscoping" />
   <parameter key="proxy-ip-addr" value="20.11.5.11" />
   <parameter key="target-ip-addr" value="${ipaddr}" />
 </service>
 <downtime interval="30000" begin="0" end="300000" /><!-- 30s, 0, 5m -->
 <downtime interval="300000" begin="300000" end="43200000" /><!-- 5m, 5m, 12h -->
 <downtime interval="600000" begin="43200000" end="432000000" /><!-- 10m, 12h, 5d -->
 <downtime begin="432000000" delete="true" /><!-- anything after 5 days delete -->
</package>
<monitor service="BarCentral" class-name=</pre>
"org.opennms.netmgt.poller.monitors.CiscoPingMibMonitor" />
```

4.6.6. CitrixMonitor

This monitor is used to test if a Citrix® Server or XenApp Server® is providing the *Independent Computing Architecture (ICA)* protocol on TCP 1494. The monitor opens a TCP socket and tests the greeting banner returns with ICA, otherwise the service is unavailable.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.CitrixMonitor |
|----------------|--|
| Remote Enabled | true |

Configuration and Usage

Table 18. Monitor specific parameters for the CitrixMonitor

| Param eter | Description | Requi red | Default value |
|---------------|---|--------------|------------------|
| retry | Amount of attempts opening a connection and try to get the greeting banner before the service goes down | option al | 0 |
| timeout | Time to wait retrieving the greeting banner ICA from TCP connection before trying a next attempt. | option al | 3000 ms |

| Param | Description | Requi | Default |
|-------|--|--------------|---------|
| eter | | red | value |
| port | TCP port where the <i>ICA</i> protocol is listening. | option al | 1494 |



If you have configure the *Metaframe Presentation Server Client* using *Session Reliability*, the TCP port is 2598 instead of 1494. You can find additional information on CTX104147. It is not verified if the monitor works in this case.

Examples

The following example configures OpenNMS Meridian to monitor the ICA protocol on TCP 1494 with 2 retries and waiting 5 seconds for each retry.

4.6.7. DhcpMonitor

This monitor is used to monitor the availability and functionality of DHCP servers. This monitor has two parts, the first one is the monitor class *DhcpMonitor* executed by *Pollerd* and the second part is a background daemon *Dhcpd* running inside the OpenNMS Meridian JVM and listening for DHCP responses. A DHCP server is tested by sending a *DISCOVER* message. If the DHCP server responds with an *OFFER* the service is marked as up. The *Dhcpd* background daemon is disabled by default and has to be activated in service-configuration.xml in OpenNMS Meridian by setting service enabled="true". The behavior for testing the DHCP server can be modified in the dhcp-configuration.xml configuration file.



It is required to install the opennms-plugin-protocol-dhcp before you can use this feature.

Installing the opennms-plugin-protocol-dhcp package

```
{apt-get,yum} install {opennms-package-base-name}-plugin-protocol-dhcp
```

If you try to start OpenNMS Meridian without the *opennms-plugin-protocol-dhcp* you will see the following error message in output.log:

An error occurred while attempting to start the "OpenNMS:Name=Dhcpd" service (class org.opennms.netmgt.dhcpd.jmx.Dhcpd). Shutting down and exiting. java.lang.ClassNotFoundException: org.opennms.netmgt.dhcpd.jmx.Dhcpd



Make sure no DHCP client is running on the OpenNMS Meridian server and using port UDP/68. If UDP/68 is already in use, you will find an error message in the manager.log. You can test if a process is listening on udp/68 with sudo ss -lnpu sport = :68.

Monitor facts

| Class Name | org.opennms.protocols.dhcp.monitor.DhcpMonitor |
|----------------|--|
| Remote Enabled | false |

Table 19. Service monitor parameters configured in poller-configuration.xml

| Paramet er | Description | Requi red | Default value |
|------------------------|--|--------------|---|
| retry | Number of retries before the service is marked as down | option al | 0 |
| timeout | Time in milliseconds to wait for the DHCP response from the server | option al | 3000 |
| rrd- repositor y | The location to write RRD data. Generally, you will not want to change this from default | option al | <pre>\$OPENNMS_HOME/share/r rd/response</pre> |
| rrd-base- name | The name of the RRD file to write (minus the extension, .rrd or .jrb) | option al | dhcp |
| ds-name | This is the name as reference for this particular data source in the RRD file | option al | dhcp |

Dhcpd configuration

Table 20. Dhcpd parameters in dhcp-configuration.xml.

| Parameter | Description | Requir ed | Default value |
|------------|--|--------------|-----------------------|
| port | Defines the port your dhcp server is using | requir ed | 5818 |
| macAddress | The MAC address which OpenNMS Meridian uses for a dhcp request | requir ed | 00:06:0D:BE :9C:B2 |

| myIpAddress | This parameter will usually be set to the IP address of the OpenNMS Meridian server, which puts the DHCP poller in relay mode as opposed to broadcast mode. In relay mode, the DHCP server being polled will unicast its responses directly back to the IP address specified by myIpAddress rather than broadcasting its responses. This allows DHCP servers to be polled even though they are not on the same subnet as the OpenNMS Meridian server, and without the aid of an external relay. Usage: myIpAddress="10.11.12.13" or myIpAddress="broadcast" | requir ed | broadcast |
|----------------------|--|--------------|-----------|
| extendedMod e | When extendedMode is false, the DHCP poller will send a DISCOVER and expect an OFFER in return. When extendedMode is true, the DHCP poller will first send a DISCOVER. If no valid response is received it will send an INFORM. If no valid response is received it will then send a REQUEST. OFFER, ACK, and NAK are all considered valid responses in extendedMode. Usage: extendedMode="true" or extendedMode="false" | requir ed | false |
| requestIpAd dress | This parameter only applies to REQUEST queries sent to the DHCP server when extendedMode is true. If an IP address is specified, that IP address will be requested in the query. If targetHost is specified, the DHCP server's own IP address will be requested. Since a well-managed server will probably not respond to a request for its own IP, this parameter can also be set to targetSubnet. This is similar to targetHost except the DHCP server's IP address is incremented or decremented by 1 to obtain an ip address that is on the same subnet. (The resulting address will not be on the same subnet if the DHCP server's subnet is a /32 or /31. Otherwise, the algorithm used should be reliable.) Usage: requestIpAddress="targetHost" or requestIpAddress="targetHost" or requestIpAddress="targetSubnet" | requir | false |

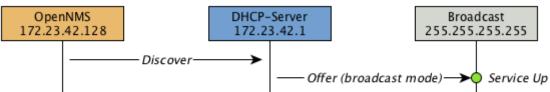


Figure 14. Visualization of DHCP message flow in broadcast mode

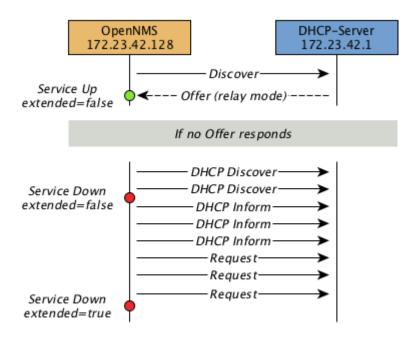


Figure 15. Visualization of DHCP message flow in relay mode

Example testing DHCP server in the same subnet

Example configuration how to configure the monitor in the poller-configuration.xml. The monitor will try to send in maximum 3 *DISCOVER* messages and waits 3 seconds for the DHCP server *OFFER* message.

Step 1: Configure a DHCP service in poller-configuration.xml

```
<service name="DHCP" 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="dhcp" />
  <parameter key="ds-name" value="dhcp" />
  </service>

<monitor service="DHCP" class-name="org.opennms.protocols.dhcp.monitor.DhcpMonitor"/>
```

Step 2: Enable the OpenNMS Meridian Dhcpd daemon in service-configuration.xml

Step 3: Configure Dhcpd to test a DHCP server in the same subnet as the OpenNMS Meridian server.

```
<DhcpdConfiguration
    port="5818"
    macAddress="00:06:0D:BE:9C:B2"
    myIpAddress="broadcast
    extendedMode="false"
    requestIpAddress="127.0.0.1">
</DhcpdConfiguration>
```

Example testing DHCP server in a different subnet in extended mode

You can use the same monitor in poller-configuration.xml as in the example above.

Configure Dhcpd to test DHCP server in a different subnet. The OFFER from the DHCP server is sent to myIpAddress.

```
<DhcpdConfiguration
    port="5818"
    macAddress="00:06:0D:BE:9C:B2"
    myIpAddress="10.4.1.234"
    extendedMode="true"
    requestIpAddress="targetSubnet">
</DhcpdConfiguration>
```



If in extendedMode, the time required to complete the poll for an unresponsive node is increased by a factor of 3. Thus it is a good idea to limit the number of retries to a small number.

4.6.8. DiskUsageMonitor

The DiskUsageMonitor monitor can be used to test the amount of free space available on certain storages of a node.

The monitor gets information about the available free storage spaces available by inspecting the *hrStorageTable* of the HOST-RESOURCES-MIB.

A storage's description (as found in the corresponding hrStorageDescr object) must match the criteria specified by the disk and match-type parameters to be monitored.

A storage's available free space is calculated using the corresponding *hrStorageSize* and *hrStorageUsed* objects.



The *hrStorageUsed* doesn't account for filesystem reserved blocks (i.e. for the super-user), so DiskUsageMonitor will report the service as unavailable only when the amount of free disk space is actually lower than free minus the percentage of reserved filesystem blocks.

This monitor uses *SNMP* to accomplish its work. Therefore systems against which it is to be used must have an SNMP agent supporting the *HOST-RESOURCES-MIB* installed and configured. Most modern *SNMP agents*, including most distributions of the *Net-SNMP agent* and the *SNMP service* that ships with *Microsoft Windows*, support this *MIB*. Out-of-box support for *HOST-RESOURCES-MIB* among commercial *Unix* operating systems may be somewhat spotty.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.DiskUsageMonitor |
|----------------|---|
| Remote Enabled | false, relies on SNMP configuration. |

Table 21. Monitor specific parameters for the DiskUsageMonitor

| Param eter | Description | Requi red | Default value |
|----------------|---|--------------|--------------------------|
| disk | A pattern that a storage's description (hrStorageDescr) must match to be taken into account. | requir ed | - |
| free | The minimum amount of free space that storages matching the criteria must have available. This parameter is evaluated as a percent of the storage's reported maximum capacity. | option al | 15 |
| match- type | The way how the pattern specified by the disk parameter must be compared to storages description Must be one of the following symbolic operators: endswith: The disk parameter's value is evaluated as a string that storages' description must end with; exact: The disk parameter's value is evaluated as a string that storages" description must exactly match; regex: The disk parameter's value is evaluated as a regular expression that storages' description must match; startswith: The disk parameter's value is evaluated as a string that storages' description must start with. Note: Comparisons are case-sensitive | option al | exact |
| port | Destination port where the SNMP requests shall be sent. | option al | from snmp- config.xml |
| retries | Deprecated. Same as retry. Parameter retry takes precedence when both are set. | option al | from snmp- config.xml |
| retry | Number of polls to attempt. | option al | from snmp- config.xml |
| timeout | Timeout in milliseconds for retrieving the values. | option al | from snmp- config.xml |

Examples

DiskUsageMonitor vs thresholds

Storages' available free space can also be monitored using thresholds if you are already collecting these data.

4.6.9. DnsMonitor

This monitor is build to test the availability of the *DNS service* on remote IP interfaces. The monitor tests the service availability by sending a DNS query for A resource record types against the DNS server to test.

The monitor is marked as *up* if the *DNS Server* is able to send a valid response to the monitor. For multiple records it is possible to test if the number of responses are within a given boundary.

The monitor can be simulated with the command line tool host:

```
~ % host -v -t a www.google.com 8.8.8.8
Trying "www.google.com"
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9324
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.google.com.
                          IN A
;; ANSWER SECTION:
www.google.com.
                  283 IN A
                               74.125.232.17
www.google.com.
                   283 IN A 74.125.232.20
                   283 IN A 74.125.232.19
www.google.com.
www.google.com.
                  283 IN A 74.125.232.16
www.google.com.
                   283 IN A
                               74.125.232.18
Received 112 bytes from 8.8.8.8#53 in 41 ms
```

TIP: This monitor is intended for testing the availability of a DNS service. If you want to monitor the DNS resolution of some of your nodes from a client's perspective, please use the DNSResolutionMonitor.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.DnsMonitor |
|----------------|---|
| Remote Enabled | true |

Table 22. Monitor specific parameters for the DnsMonitor

| Parameter | Description | Requi red | Default value |
|-----------|---|--------------|------------------|
| retry | Number of retries before the service is marked as down | option al | 0 |
| timeout | Time in milliseconds to wait for the <i>A Record</i> response from the server | option al | 5000 |
| port | UDP Port for the DNS server | option al | 53 |
| lookup | DNS A Record for lookup test | option al | localhost |

| Parameter | Description | Requi red | Default value |
|------------------------------|--|--------------|------------------|
| fatal- response- codes | A comma-separated list of numeric DNS response codes that will be considered fatal if present in the server's response. Default value is 2 corresponds to <i>Server Failed</i> . A list of codes and their meanings is found in RFC 2929 | option al | 2 |
| min-answers | Minmal number of records in the DNS server respone for the given lookup | option al | - |
| max-answers | Maximal number of records in the DNS server respone for the given lookup | option al | - |

Examples

The given examples shows how to monitor if the IP interface from a given DNS server resolves a DNS request. This service should be bound to a DNS server which should be able to give a valid DNS respone for DNS request *www.google.com*. The service is *up* if the DNS server gives between 1 and 10 *A record* responses.

Example configuration monitoring DNS request for a given server for www.google.com

4.6.10. DNSResolutionMonitor

The DNS resolution monitor, tests if the node label of an OpenNMS Meridian node can be resolved. This monitor uses the name resolver configuration from the poller configuration or from the operating system where OpenNMS Meridian is running on. It can be used to test a client behavior for a given host name. For example: Create a node with the node label www.google.com and an IP interface. Assigning the DNS resolution monitor on the IP interface will test if www.google.com can be resolved using the DNS configuration defined by the poller. The response from the A record lookup can be any address, it is not verified with the IP address on the OpenNMS Meridian IP interface where the monitor is assigned to.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.DNSResolutionMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 23. Monitor specific parameters for the DNSResolutionMonitor

| Parame ter | Description | Requ ired | Default value |
|---------------------|---|--------------|---|
| resoluti on-type | Type of record for the node label test. Allowed values v4 for A records, v6 for AAAA record, both A and AAAA record must be available, either A or AAAA record must be available. | optio nal | either |
| nameserv er | The DNS server to query for the records. | optio nal | Use the servers defined by the system running OpenNMS Meridian |
| retry | Amount of attempts to resolve the node label before the service goes down | requi red | - |
| timeout | Time to wait for a <i>A</i> and/or <i>AAAA record</i> from the system configured <i>DNS server</i> before trying a next attempt. | requi red | - |

Examples

The following example shows the possibilities monitoring IPv4 and/or IPv6 for the service configuration:

```
<!-- Assigned service test if the node label is resolved for an A record -->
<service name="DNS-Resolution-v4" interval="300000" user-defined="false" status="on">
   <parameter key="retry" value="2"/>
   <parameter key="timeout" value="2000"/>
   <parameter key="resolution-type" value="v4"/>
   <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
   <parameter key="rrd-base-name" value="dns-res-v4"/>
   <parameter key="ds-name" value="dns-res-v4"/>
</service>
<!-- Assigned service test if the node label is resolved for an AAAA record using a
specific DNS server -->
<service name="DNS-Resolution-v6" interval="300000" user-defined="false" status="on">
   <parameter key="retry" value="2"/>
   <parameter key="timeout" value="2000"/>
   <parameter key="resolution-type" value="v6"/>
   <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
   <parameter key="rrd-base-name" value="dns-res-v6"/>
   <parameter key="ds-name" value="dns-res-v6"/>
   <parameter key="nameserver" value="8.8.8.8"/>
</service>
<!-- Assigned service test if the node label is resolved for an AAAA record AND A
```

```
record -->
<service name="DNS-Resolution-v4-and-v6" interval="300000" user-defined="false"</pre>
status="on">
    <parameter key="retry" value="2"/>
    <parameter key="timeout" value="2000"/>
    <parameter key="resolution-type" value="both"/>
    <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
    <parameter key="rrd-base-name" value="dns-res-both"/>
    <parameter key="ds-name" value="dns-res-both"/>
</service>
<!-- Assigned service test if the node label is resolved for an AAAA record OR A
record -->
<service name="DNS-Resolution-v4-or-v6" interval="300000" user-defined="false" status</pre>
="on">
    <parameter key="retry" value="2"/>
    <parameter key="timeout" value="2000"/>
    <parameter key="resolution-type" value="either"/>
    <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
    <parameter key="rrd-base-name" value="dns-res-either"/>
    <parameter key="ds-name" value="dns-res-either"/>
</service>
<monitor service="DNS-Resolution-v4" class-name=</pre>
"org.opennms.netmgt.poller.monitors.DNSResolutionMonitor" />
<monitor service="DNS-Resolution-v6" class-name=</pre>
"org.opennms.netmgt.poller.monitors.DNSResolutionMonitor" />
<monitor service="DNS-Resolution-v4-and-v6" class-name=</pre>
"org.opennms.netmgt.poller.monitors.DNSResolutionMonitor" />
<monitor service="DNS-Resolution-v4-or-v6" class-name=</pre>
"org.opennms.netmgt.poller.monitors.DNSResolutionMonitor" />
```

To have response time graphs for the name resolution you have to configure RRD graphs for the given ds-names (dns-res-v4, dns-res-v6, dns-res-both, dns-res-either) in '\$OPENNMS HOME/etc/response-graph.properties'.

DNSResolutionMonitor vs DnsMonitor

The DNSResolutionMonitor is used to measure the availability and record outages of a name resolution from client perspective. The service is mainly used for websites or similar public available resources. It can be used in combination with the Page Sequence Monitor to give a hint if a website isn't available for DNS reasons.

The DnsMonitor on the other hand is a test against a specific DNS server. In OpenNMS Meridian the DNS server is the node and the DnsMonitor will send a lookup request for a given A record to the DNS server IP address. The service goes down if the DNS server doesn't have a valid A record in his zone database or as some other issues resolving A records.

4.6.11. FtpMonitor

The FtpMonitor is able to validate ftp connection dial-up processes. The monitor can test ftp server on multiple ports and specific login data.

The service using the FtpMonitor is up if the FTP server responds with return codes between 200 and 299. For special cases the service is also marked as up for 425 and 530.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.FtpMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 24. Monitor specific parameters for the FtpMonitor.

| Param eter | Description | Requi red | Default value |
|---------------|---|--------------|------------------|
| retry | Number of attempts to get a valid FTP response/response-text | option al | 0 |
| timeout | Timeout in milliseconds for TCP connection establishment. | option al | 3000 |
| port | A list of TCP ports to which connection shall be tried. | option al | 20,21 |
| passwor d | This parameter is meant to be used together with the user parameter to perform basic authentication. This parameter specify to password to be used. The user and password parameters are ignored when the basic-authentication parameter is defined. | option al | empty string |
| userid | This parameter is meant to be used together with the password parameter to perform basic authentication. This parameter specify to user ID to be used. The userid and password parameters are ignored when the basic-authentication parameter is defined. | option al | _ |

Examples

Some example configuration how to configure the monitor in the 'poller-configuration.xml'

```
<service name="FTP" interval="300000" user-defined="false" status="on">
<parameter key="retry" value="1"/>
<parameter key="timeout" value="3000"/>
<parameter key="port" value="21"/>
<parameter key="userid" value=""/>
<parameter key="password" value=""/>
</service>
<service name="FTP-Customer" interval="300000" user-defined="false" status="on">
<parameter key="retry" value="1"/>
<parameter key="timeout" value="3000"/>
<parameter key="port" value="21"/>
<parameter key="userid" value="Customer"/>
<parameter key="password" value="MySecretPassword"/>
<monitor service="FTP" class-name="org.opennms.netmgt.poller.monitors.FtpMonitor"/>
<monitor service="FTP-Customer" class-name=</pre>
"org.opennms.netmgt.poller.monitors.FtpMonitor"/>
```

Hint

Comment from FtpMonitor source

Also want to accept the following ERROR message generated by some FTP servers following a QUIT command without a previous successful login: "530 QUIT: User not logged in. Please login with USER and PASS first."

Also want to accept the following ERROR message generated by some FTP servers following a QUIT command without a previously successful login: "425 Session is disconnected."

See also: http://tools.ietf.org/html/rfc959

4.6.12. HostResourceSwRunMonitor

This monitor test the running state of one or more processes. It does this via SNMP by inspecting the *hrSwRunTable* of the HOST-RESOURCES-MIB. The test is done by matching a given process as *hrSwRunName* against the numeric value of the *hrSwRunState*.

This monitor uses *SNMP* to accomplish its work. Therefore systems against which it is to be used must have an SNMP agent installed and configured. Furthermore, the *SNMP agent* on the system must support the *HOST-RESOURCES-MIB*. Most modern *SNMP agents*, including most distributions of the *Net-SNMP agent* and the *SNMP service* that ships with *Microsoft Windows*, support this *MIB*. Out-of-box support for *HOST-RESOURCES-MIB* among commercial *Unix* operating systems may be somewhat spotty.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.HostResourceSwRunMonitor |
|------------|---|
|------------|---|

| е | | |
|---|--|--|
|---|--|--|

Configuration and Usage

 ${\it Table~25.}\ Monitor\ specific\ parameters\ for\ the\ HostResource SwRunMonitor$

| Parameter | Description | Requi red | Default value |
|------------------------|---|--------------|-----------------------------|
| port | The port of the SNMP agent of the server to test. | option al | from snmp- config.xml |
| retry | Number of attempts to get a valid response before marking the service as <i>down</i> . | option al | from snmp- config.xml |
| timeout | Timeout in milliseonds wating for the <i>SNMP response</i> for the process run state from the agent. | option al | from snmp- config.xml |
| service- name | The name of the process to be monitored. This parameter's value is case-sensitive and is evaluated as an exact match. | requir ed | - |
| match-all | If the process name appears multiple times in the <i>hrSwRunTable</i> , and this parameter is set to true, then all instances of the named process must match the value specified for run-level. | option al | false |
| run-level | The maximum allowable value of hrSWRunStatus among running(1), runnable(2) = waiting for resource notRunnable(3) = loaded but waiting for event invalid(4) = not loaded | option al | 2 |
| service- name-oid | The numeric object identifier (OID) from which process names are queried. Defaults to hrSwRunName and should never be changed under normal circumstances. That said, changing it to hrSwRunParameters (.1.3.6.1.2.1.25.4.2.1.5) is often helpful when dealing with processes running under Java Virtual Machines which all have the same process name java. | option al | .1.3.6.1.2.1.2 5.4.2.1.2 |
| service- status-oid | The numeric object identifier (OID) from which run status is queried. Defaults to | | |

Examples

The following example shows how to monitor the process called *httpd* running on a server using this monitor. The configuration in poller-configuration.xml has to be defined as the following:

- 1 Name of the process on the system
- 2 Test the state if the process is in a valid state, i.e. have a run-level no higher than notRunnable(3)
- ③ If the *httpd* process runs multiple times the test is done for each instance of the process.

4.6.13. HttpMonitor

The HTTP monitor tests the response of an HTTP server on a specific HTTP 'GET' command. During the poll, an attempt is made to connect on the specified port(s). The monitor can test web server on multiple ports. By default the a test is made against port 80, 8080 and 8888. If the connection request is successful, an HTTP 'GET' command is sent to the interface. The response is parsed and a return code extracted and verified.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.HttpMonitor |
|----------------|--|
| Remote Enabled | true |

Table 26. Monitor specific parameters for the HttpMonitor

| Paramet er | Description | Req uire d | Default value |
|------------------------------|--|------------------|---------------|
| basic- authentic ation | 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. | opti onal | |

| Paramet er | Description | Req uire d | Default value |
|-----------------------------|--|------------------|--|
| 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. | opti onal | _ |
| host-name | Specify the <i>Host</i> header's value. | opti onal | _ |
| nodelabel -host -name | If the host-name parameter isn't set and the resolve-ip parameter is set to false, then OpenNMS Meridian will use the node's label to set the <i>Host</i> header's value if this parameter is set to true. Otherwise, OpenNMS Meridian will fall back using the node interface's IP address as <i>Host</i> header value. | opti onal | false |
| password | This parameter is meant to be used together with the user parameter to perform basic authentication. This parameter specify to password to be used. The user and password parameters are ignored when the basic-authentication parameter is defined. | opti onal | empty string |
| port | A list of TCP ports to which connection shall be tried. | opti onal | 80,8080,8888 |
| retry | Number of attempts to get a valid HTTP response/response-text | opti onal | 0 |
| resolve-ip | If the host-name parameter isn't set and this parameter is set to true, OpenNMS Meridian will use DNS to resolve the node interface's IP address, and use the result to set the Host header's value. When set to false and the host-name parameter isn't set, OpenNMS Meridian will try to use the nodelabel-host-name parameter to set the Host header's value. | opti onal | false |
| response | A comma-separated list of acceptable HTTP response code ranges. Example: 200-202,299 | _ | If the url parameter is set to /, the default value for this parameter is 100-499, otherwise it's 100-399. |

| Paramet er | Description | Req uire d | Default value |
|-------------------|--|------------------|---------------------|
| response- text | Text to look for in the response body. This will be matched against every line, and it will be considered a success at the first match. If there is a ~ at the beginning of the parameter, the rest of the string will be used as a regular expression pattern match, otherwise the match will be a substring match. The regular expression match is anchored at the beginning and end of the line, so you will likely need to put a .* on both sides of your pattern unless you are going to be matching on the entire line. | opti onal | |
| timeout | Timeout in milliseconds for TCP connection establishment. | opti onal | 3000 |
| url | URL to be retrieved via the HTTP 'GET' command | opti onal | / |
| user | This parameter is meant to be used together with the password parameter to perform basic authentication. This parameter specify to user ID to be used. The user and password parameters are ignored when the basic-authentication parameter is defined. | opti onal | - |
| user- agent | Allows you to set the <i>User-Agent</i> HTTP header (see also RFC2616 section 14.43). | opti onal | OpenNMS HttpMonitor |
| verbose | When set to <i>true</i> , full communication between client and the webserver will be logged (with a log level of DEBUG). | opti onal | - |

Examples

```
<!-- Test HTTP service on port 80 only -->
<service name="HTTP" interval="300000" user-defined="false" status="on">
 <parameter key="retry" value="2"/>
 <parameter key="timeout" value="3000"/>
 <parameter key="port" value="80"/>
 <parameter key="url" value="/"/>
</service>
<!-- Test for virtual host opennms.com running -->
<service name="OpenNMSdotCom" interval="300000" user-defined="false" status="on">
 <parameter key="retry" value="1"/>
 <parameter key="timeout" value="3000"/>
 <parameter key="port" value="80"/>
 <parameter key="host-name" value="opennms.com"/>
 <parameter key="url" value="/solutions"/>
 <parameter key="response" value="200-202,299"/>
 <parameter key="response-text" value="~.*[Cc]onsulting.*"/>
</service>
<!-- Test for instance of OpenNMS 1.2.9 running -->
<service name="OpenNMS-129" interval="300000" user-defined="false" status="on">
 <parameter key="retry" value="1"/>
 <parameter key="timeout" value="3000"/>
 <parameter key="port" value="8080"/>
 <parameter key="url" value="/opennms/event/list"/>
 <parameter kev="basic-authentication" value="admin:admin"/>
 <parameter key="response" value="200"/>
</service>
<monitor service="HTTP" class-name="org.opennms.netmgt.poller.monitors.HttpMonitor" />
<monitor service="OpenNMSdotCom" class-name=</pre>
"org.opennms.netmgt.poller.monitors.HttpMonitor" />
<monitor service="OpenNMS-129" class-name=</pre>
"org.opennms.netmgt.poller.monitors.HttpMonitor" />
```

Testing filtering proxies with HttpMonitor

If you have a filtering proxy server that is supposed to allow retrieval of some URLs but deny others, you can use the HttpMonitor to verify this behavior.

Let's say that our proxy server is running on TCP port 3128, and that we should always be able to retrieve http://www.opennms.org/ but never http://www.myspace.com/ (hey, this is a workplace after all!). To test this behaviour, one could create the following service monitors:

```
<service name="HTTP-Allow-opennms.org" interval="300000" user-defined="false" status=</pre>
"on">
 <parameter key="retry" value="1"/>
 <parameter key="timeout" value="3000"/>
 <parameter key="port" value="3128"/>
 <parameter key="url" value="http://www.opennms.org/"/>
 <parameter key="response" value="200-399"/>
</service>
<service name="HTTP-Block-myspace.com" interval="300000" user-defined="false" status=</pre>
 <parameter key="retry" value="1"/>
 <parameter key="timeout" value="3000"/>
 <parameter key="port" value="3128"/>
 <parameter key="url" value="http://www.myspace.com/"/>
 <parameter key="response" value="400-599"/>
</service>
<monitor service="HTTP-Allow-opennms.org" class-name=</pre>
"org.opennms.netmgt.poller.monitors.HttpMonitor"/>
<monitor service="HTTP-Block-myspace.com" class-name=</pre>
"org.opennms.netmgt.poller.monitors.HttpMonitor"/>
```

4.6.14. HttpPostMonitor

If it is required to *HTTP POST* any arbitrary content to a remote *URI*, the HttpPostMonitor can be used. A use case is to HTTP POST to a SOAP endpoint.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.HttpPostMonitor |
|----------------|--|
| Remote Enabled | false |

Table 27. Monitor specific parameters for the HttpPostMonitor

| Paramet er | Description | Requi red | Default value |
|-------------------|---|--------------|------------------|
| payload | The body of the POST, for example properly escaped XML. | requir ed | - |
| auth- password | The password to use for HTTP BASIC auth. | option al | - |
| auth- username | The username to use for HTTP BASIC auth. | option al | _ |

| Paramet er | Description | Requi red | Default value |
|------------------|---|--------------|------------------|
| banner | A string that is matched against the response of the HTTP POST. If the output contains the banner, the service is determined as up. Specify a regex by starting with ~. | option al | - |
| charset | Set the character set for the POST. | option al | UTF-8 |
| mimetype | Set the mimetype for the POST. | option al | text/xml |
| port | The port for the web server where the POST is send to. | option al | 80 |
| scheme | The connection scheme to use. | option al | http |
| usesslfil ter | Enables or disables the SSL ceritificate validation. true - false | option al | false |
| uri | The uri to use during the POST. | option al | / |

Examples

The following example would create a POST that contains the payload Word.

```
<service name="MyServlet" interval="300000" user-defined="false" status="on">
    <parameter key="banner" value="Hello"/>
    <parameter key="port" value="8080"/>
    <parameter key="uri" value="/MyServlet">
        <parameter key="payload" value="World"/>
        <parameter key="retry" value="1"/>
        <parameter key="timeout" value="30000"/>
        </service>
    <monitor service="MyServlet" class-name=
        "org.opennms.netmgt.poller.monitors.HttpPostMonitor"/>
```

The resulting POST looks like this:

```
POST /MyServlet HTTP/1.1
Content-Type: text/xml; charset=utf-8
Host: <ip_addr_of_interface>:8080
Connection: Keep-Alive
World
```

4.6.15. HttpsMonitor

The HTTPS monitor tests the response of an SSL-enabled HTTP server. The HTTPS monitor is an SSL-enabled extension of the HTTP monitor with a default TCP port value of 443. All HttpMonitor parameters apply, so please refer to HttpMonitor's documentation for more information.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.HttpsMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 28. Monitor specific parameters for the HttpsMonitor

| Paramete | Description | Require | Default |
|----------|---|----------|---------|
| r | | d | value |
| port | A list of TCP ports to which connection shall be tried. | optional | 443 |

Examples

4.6.16. IcmpMonitor

The ICMP monitor tests for ICMP service availability by sending *echo request* ICMP messages. The service is considered available when the node sends back an *echo reply* ICMP message within the specified amount of time.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.IcmpMonitor |
|----------------|--|
| Remote Enabled | true with some restrictions (see below) |

Configuration and Usage

Table 29. Monitor specific parameters for the IcmpMonitor

| Parameter | Description | Require d | Default value |
|--------------------------|---|--------------|------------------|
| retry | Number of attempts to get a response. | optional | 2 |
| timeout | Time in milliseconds to wait for a response. | optional | 800 |
| allow- fragmentation | Whether to set the "Don't Fragment" bit on outgoing packets | optional | true |
| dscp | DSCP traffic-control value. | optional | 0 |
| packet-size | Number of bytes of the ICMP packet to send. | optional | 64 |
| thresholding- enabled | Enables ICMP thresholding. | optional | true |

Examples

Note on Remote Poller

The IcmpMonitor needs the JNA ICMP implementation to function on remote poller. Though, corner cases exist where the IcmpMonitor monitor won't work on remote poller. Examples of such corner cases are: Windows when the remote poller isn't running has administrator, and Linux on ARM / Rasperry Pi. JNA is the default ICMP implementation used in the remote poller.

4.6.17. ImapMonitor

This monitor checks if an IMAP server is functional. The test is done by initializing a very simple IMAP conversation. The ImapMonitor establishes a TCP connection, sends a logout command and test the IMAP server responses.

The behavior can be simulated with telnet:

```
telnet mail.myserver.de 143
Trying 62.108.41.197...
Connected to mail.myserver.de.
Escape character is '^]'.

* OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE STARTTLS LOGINDISABLED] Dovecot ready. ①
ONMSPOLLER LOGOUT ②

* BYE Logging out ③
ONMSPOLLER OK Logout completed.
Connection closed by foreign host.
```

- 1 Test IMAP server banner, it has to start * OK to be up
- ② Sending a ONMSPOLLER LOGOUT
- 3 Test server responds with, it has to start with * BYE to be up

If one of the tests in the sample above fails the service is marked *down*.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.ImapMonitor |
|----------------|--|
| Remote Enabled | false |

Configuration and Usage

Table 30. Monitor specific parameters for the ImapMonitor

| Param eter | Description | Requi red | Default value |
|---------------|--|--------------|------------------|
| retry | Number of attempts to get a valid IMAP response | option al | 0 |
| timeout | Time in milliseconds to wait retrieving the banner from TCP connection before trying a next attempt. | option al | 3000 |
| port | The port of the IMAP server. | option al | 143 |

Examples

Some example configuration how to configure the monitor in the poller-configuration.xml

4.6.18. ImapsMonitor

The IMAPS monitor tests the response of an SSL-enabled IMAP server. The IMAPS monitor is an SSL-enabled extension of the IMAP monitor with a default TCP port value of 993. All ImapMonitor parameters apply, so please refer to ImapMonitor's documentation for more information.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.ImapsMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 31. Monitor specific parameters for the ImapsMonitor

| Paramete | Description | Require | Default |
|----------|--|----------|---------|
| r | | d | value |
| port | The destination port where connections shall be attempted. | optional | 993 |

Examples

4.6.19. JCifsMonitor

This monitor allows to test a file sharing service based on the CIFS/SMB protocol.



This monitor is not installed by default. You have to install opennmms-plugin-protocol-cifs from your OpenNMS Meridian installation repository.

With the JCIFS monitor you have different possibilities to test the availability of the JCIFS service:

With the *JCifsMonitor* it is possible to run tests for the following use cases:

- share is available in the network
- a given file exists in the share
- a given folder exists in the share
- a given folder should contain at least one (1) file
- a given folder folder should contain no (0) files
- by testing on files and folders, you can use a regular expression to ignore specific file and folder names from the test

A network resource in SMB like a file or folder is addressed as a UNC Path.

\\server\share\folder\file.txt

The Java implementation *jCIFS*, which implements the *CIFS/SMB* network protocol, uses *SMB* URLs to access the network resource. The same resource as in our example would look like this as an SMB URL:

smb://workgroup;user:password@server/share/folder/file.txt

The *JCifsMonitor* can **not** test:

- file contains specific content
- a specific number of files in a folder, for example folder should contain exactly / more or less than x files
- Age or modification time stamps of files or folders
- Permissions or other attributes of files or folders

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.JCifsMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

Table 32. Monitor specific parameters for the JCifsMonitor

| Parameter | Description | Requi red | Default value |
|-----------------------|---|--------------|------------------|
| retry | Number of retries before the service is marked as <i>down</i> . | option al | 0 |
| timeout | Time in milliseconds to wait for the SMB service. | option al | 3000 |
| domain | Windows domain where the user is located. You don't have to use the domain parameter if you use local user accounts. | option al | empty String |
| username | Username to access the resource over a network | option al | empty String |
| password | Password for the user | option al | empty String |
| path | Path to the resource you want to test | requir ed | empty String |
| mode | The test mode which has the following options path_exist: Service is <i>up</i> if the resource is accessible path_not_exist: Service is <i>up</i> if the resource is not accessible folder_empty: Service is <i>up</i> if the folder is empty (0 files) folder_not_empty: Service is <i>up</i> if the folder has at least one file | option al | path_exis t |
| smbHost | Override the IP address of the SMB url to check shares on different file servers. | option al | empty String |
| folderIgnore Files | Ignore specific files in folder with regular expression. This parameter will just be applied on folder_empty and folder_not_empty, otherwise it will be ignored. | option al | - |



It makes little sense to have retries higher than 1. It is a waste of resources during the monitoring.



Please consider, if you are accessing shares with Mac OSX you have some side effects with the hidden file '.DS_Store.' It could give you false positives in monitoring, you can use then the folderIgnoreFiles parameter.

Example test existence of a file

This example shows how to configure the *JCifsMonitor* to test if a file share is available over a network. For this example we have access to a share for error logs and we want to get an outage if we have any error log files in our folder. The share is named *log*. The service should go back to normal if the error log file is deleted and the folder is empty.

JCifsMonitor configuration to test that a shared folder is empty

- 1 Name of the SMB or Microsoft Windows Domain
- 2 User for accessing the share
- 3 Password for accessing the share
- 4 Path to the folder inside of the share as part of the SMB URL
- 5 Mode is set to folder empty

4.6.20. JDBCMonitor

The *JDBCMonitor* checks that it is able to connect to a database and checks if it is able to get the database catalog from that database management system (DBMS). It is based on the JDBC technology to connect and communicate with the database.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.JDBCMonitor |
|----------------|--|
| Remote Enabled | true |

Table 33. Monitor specific parameters for the JDBCMonitor

| Parame ter | Description | Requir ed | Default value |
|---------------|--------------------------|--------------|--|
| driver | JDBC driver class to use | requir ed | com.sybase.jdbc2.jdbc.SybDriver |
| url | JDBC Url to connect to. | requir ed | <pre>jdbc:sybase:Tds:OPENNMS_JDBC_HOS TNAME/tempdb</pre> |
| user | Database user | requir ed | sa |

| Parame ter | Description | Requir ed | Default value |
|---------------|--|--------------|---------------|
| password | Database password | requir ed | empty string |
| timeout | Timeout in ms for the database connection | option al | 3000 |
| retries | How many retries should be performed before failing the test | option al | 0 |



The *OPENNMS_JDBC_HOSTNAME* is replaced in the *url* parameter with the IP or resolved hostname of the interface the monitored service is assigned to.

Provide the database driver

The *JDBCMonitor* is based on *JDBC* and requires a JDBC driver to communicate with any database. Due to the fact that OpenNMS Meridian itself uses a PostgreSQL database, the PostgreSQL JDBC driver is available out of the box. For all other database systems a compatible JDBC driver has to be provided to OpenNMS Meridian as a *jar-file*. To provide a JDBC driver place the *driver-jar* in the opennms/lib folder of your OpenNMS Meridian. To use the *JDBCMonitor* from a remote poller, the *driver-jar* has to be provided to the *Remote Poller* too. This may be tricky or impossible when using the *Java Webstart Remote Poller*, because of code signing requirements.

Examples

The following example checks if the PostgreSQL database used by OpenNMS Meridian is available.

```
<service name="OpenNMS-DBMS" interval="30000" user-defined="true" status="on">
    <parameter key="driver" value="org.postgresql.Driver"/>
     <parameter key="url" value="jdbc:postgresql://OPENNMS_JDBC_HOSTNAME:5432/opennms"/>
     <parameter key="user" value="opennms"/>
     <parameter key="password" value="opennms"/>
     </service>

<monitor service="OpenNMS-DBMS" class-name=
     "org.opennms.netmgt.poller.monitors.JDBCMonitor" />
```

4.6.21. JDBCStoredProcedureMonitor

The *JDBCStoredProcedureMonitor* checks the result of a stored procedure in a remote database. The result of the stored procedure has to be a boolean value (representing true or false). The service associated with this monitor is marked as up if the stored procedure returns true and it is marked as down in all other cases. It is based on the *JDBC* technology to connect and communicate with the database.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.JDBCStoredProcedureMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

Table 34. Monitor specific parameters for the JDBCStoredProcedureMonitor

| Parameter | Description | Requi red | Default value |
|----------------------|--|--------------|--|
| driver | JDBC driver class to use | requir ed | <pre>com.sybase.jdbc2.jdbc.SybDrive r</pre> |
| url | JDBC Url to connect to. | requir ed | <pre>jdbc:sybase:Tds:OPENNMS_JDBC_H OSTNAME/tempdb</pre> |
| user | Database user | requir ed | sa |
| password | Database password | requir ed | empty string |
| timeout | Timeout in ms for the database connection | option al | 3000 |
| retries | How many retries should be performed before failing the test | option al | 0 |
| stored- procedure | Name of the database stored procedure to call | requir ed | - |
| schema | Name of the database schema in which the stored procedure is | option al | test |



The *OPENNMS_JDBC_HOSTNAME* is replaced in the *url* parameter with the IP or resolved hostname of the interface the monitored service is assigned to.

Provide the database driver

The JDBCStoredProcedureMonitor is based on JDBC and requires a JDBC driver to communicate with any database. Due to the fact that OpenNMS Meridian itself uses a PostgreSQL database, the PostgreSQL JDBC driver is available out of the box. For all other database systems a compatible JDBC driver has to be provided to OpenNMS Meridian as a jar-file. To provide a JDBC driver place the driver-jar in the opennms/lib folder of your OpenNMS Meridian. To use the JDBCStoredProcedureMonitor from a remote poller, the driver-jar has to be provided to the Remote Poller too. This may be tricky or impossible when using the Java Webstart Remote Poller, because of code signing requirements.

Examples

The following example checks a stored procedure added to the PostgreSQL database used by

OpenNMS Meridian. The stored procedure returns true as long as less than 250000 events are in the events table of OpenNMS Meridian.

Stored procedure which is used in the monitor

```
CREATE OR REPLACE FUNCTION eventlimit_sp() RETURNS boolean AS
$BODY$DECLARE
num_events integer;
BEGIN
    SELECT COUNT(*) into num_events from events;
    RETURN num_events > 250000;
END;$BODY$
LANGUAGE plpgsql VOLATILE NOT LEAKPROOF
COST 100;
```

4.6.22. JDBCQueryMonitor

The *JDBCQueryMonitor* runs an SQL query against a database and is able to verify the result of the query. A read-only connection is used to run the SQL query, so the data in the database is not altered. It is based on the JDBC technology to connect and communicate with the database.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.JDBCQueryMonitor |
|----------------|---|
| Remote Enabled | false |

Table 35. Monitor specific parameters for the JDBCQueryMonitor

| Para meter | Description | Requ ired | Default value |
|---------------|--------------------------|--------------|---|
| driver | JDBC driver class to use | requi red | <pre>com.sybase.jdbc2.jdbc.Syb Driver</pre> |

| Para meter | Description | Requ ired | Default value |
|---------------|---|--------------|--|
| url | JDBC URL to connect to. | requi red | <pre>jdbc:sybase:Tds:OPENNMS_J DBC_HOSTNAME/tempdb</pre> |
| user | Database user | requi red | sa |
| passwo rd | Database password | requi red | empty string |
| query | The SQL query to run | requi red | - |
| action | What evaluation action to perform | requi red | row_count |
| column | The result column to evaluate against | requi red | - |
| operat or | Operator to use for the evaluation | requi red | >= |
| operan d | The operand to compare against the SQL query result | requi red | depends on the action |
| messag e | The message to use if the service is down. Both operands and the operator are added to the message too. | optio nal | generic message depending on the action |
| timeou t | Timeout in ms for the database connection | optio nal | 3000 |
| retrie s | How many retries should be performed before failing the test | optio nal | 0 |



The OPENNMS_JDBC_HOSTNAME is replaced in the url parameter with the IP or resolved hostname of the interface the monitored service is assigned to.

Table 36. Available action parameters and their default operand

| Parameter | Description | Default operand |
|--------------|--|-----------------|
| row_count | The number of returned rows is compared, not a value of the resulting rows | 1 |
| compare_stri | Strings are always checked for equality with the operand | - |
| compare_int | An integer from a column of the first result row is compared | 1 |

Table 37. Available operand parameters

| Paramete r | XML entity to use in XML configs |
|---------------|----------------------------------|
| = | = |
| < | 8lt; |
| > | > |
| != | != |
| \(\) | <= |
| >= | >= |

Evaluating the action - operator - operand

Only the first result row returned by the SQL query is evaluated. The evaluation can be against the value of one column or the number of rows returned by the SQL query.

Provide the database driver

The *JDBCQueryMonitor* is based on *JDBC* and requires a JDBC driver to communicate with any database. Due to the fact that OpenNMS Meridian itself uses a PostgreSQL database, the PostgreSQL JDBC driver is available out of the box. For all other database systems a compatible JDBC driver has to be provided to OpenNMS Meridian as a *jar-file*. To provide a JDBC driver place the *driver-jar* in the opennms/lib folder of your OpenNMS Meridian. To use the *JDBCQueryMonitor* from a remote poller, the *driver-jar* has to be provided to the *Remote Poller* too. This may be tricky or impossible when using the *Java Webstart Remote Poller*, because of code signing requirements.

Examples

The following example checks if the number of events in the OpenNMS Meridian database is fewer than 250000.

4.6.23. JmxMonitor

The JMX monitor allows to test service availability of Java applications. The monitor offers the following functionalities:

- test the application's connectivity via *JMX*
- existence of management beans
- test the status of a single or multiple management beans and evaluate their value

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.Jsr160Monitor |
|----------------|--|
| Remote Enabled | true |

Configuration and Usage

Table 38. Monitor specific parameters for the JmxMonitor

| Parameter | Description | Requi red | Default value |
|--|---|--------------|-------------------------|
| retry | Number of attempts to get a response | option al | 3 |
| timeout | Time in milliseconds to wait for a response | option al | ? |
| port | Destination port where the JMX requests shall be sent | option al | from jmx- config.xml |
| factory | Set this to PASSWORD-CLEAR if credentials are required | option al | STANDARD |
| protocol | Protocol used in the <i>JMX</i> connection string | option al | rmi |
| urlPath | Path used in <i>JMX</i> connection string | option al | /jmxrmi |
| rmiServerPo rt | RMI port | option al | 45444 |
| remoteJMX | Use an alternative JMX URL scheme | option al | false |
| beans. <vari able></vari | Defines a <i>mbeans</i> objectname to access. The ´ <variable>´ name is arbitrary.</variable> | option al | - |
| tests. <vari< td=""><td>Tests a <i>mbeans</i> attribute value. The '<variable>' name is arbitrary.</variable></td><td>option al</td><td>-</td></vari<> | Tests a <i>mbeans</i> attribute value. The ' <variable>' name is arbitrary.</variable> | option al | - |

Examples

Test a specific management bean for a value



Reserved XML characters like >, <, " need to be escaped.

4.6.24. JolokiaBeanMonitor

The JolokiaBeanMonitor is a JMX monitor specialized for the use with the Jolokia framework. If it is required to execute a method via *JMX* or poll an attribute via *JMX*, the *JolokiaBeanMonitor* can be used. It requires a fully installed and configured *Jolokia agent* to be deployed in the JVM container. If required it allows attribute names, paths, and method parameters to be provided additional arguments to the call. To determine the status of the service the *JolokiaBeanMonitor* relies on the output to be matched against a banner. If the banner is part of the output the status is interpreted as up. If the banner is not available in the output the status is determined as *down*. Banner matching supports regular expression and substring match.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.JolokiaBeanMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

Table 39. Monitor specific parameters for the JolokiaBeanMonitor

| Parame ter | Description | Required | Default value |
|-----------------------|--|---|------------------|
| beannam e | The bean name to query against. | required | - |
| attrnam e | The name of the JMX attribute to scrape. | optional (attrname or methodname must be set) | - |
| attrpat h | The attribute path. | optional | - |
| auth- usernam e | The username to use for HTTP BASIC auth. | optional | - |
| auth- passwor d | The password to use for HTTP BASIC auth. | optional | - |
| banner | A string that is match against the output of the system-call. If the output contains the banner, the service is determined as up . Specify a regex by starting with \sim . | optional | - |
| input1 | Method input | optional | - |
| input2 | Method input | optional | - |
| methodn ame | The name of the bean method to execute, output will be compared to banner. | optional (attrname or methodname must be set) | - |
| port | The port of the jolokia agent. | optional | 8080 |
| url | The jolokia agent url. Defaults to "http:// <ipaddr>:<port>/jolokia"</port></ipaddr> | optional | - |

Table 40. Variables which can be used in the configuration

| Variable | Description |
|------------|--|
| \${ipaddr} | IP-address of the interface the service is bound to. |
| \${port} | Port the service it bound to. |

Examples

Some example configuration how to configure the monitor in the poller-configuration.xml

```
<parameter key="url" value="http://${ipaddr}:${port}/jolokia"/>
<parameter key="url" value="https://${ipaddr}:${port}/jolokia"/>
```

AttrName vs MethodName

The JolokiaBeanMonitor has two modes of operation. It can either scrape an attribute from a bean, or execute a method and compare output to a banner. The method execute is useful when your application has it's own test methods that you would like to trigger via OpenNMS Meridian.

The args to execute a test method called "superTest" that take in a string as input would look like this:

```
<parameter key="beanname" value="MyBean" />
<parameter key="methodname" value="superTest" />
<parameter key="input1" value="someString"/>
```

The args to scrape an attribute from the same bean would look like this:

```
<parameter key="beanname" value="MyBean" />
<parameter key="attrname" value="upTime" />
```

4.6.25. LdapMonitor

The LDAP monitor tests for LDAP service availability. The LDAP monitor first tries to establish a TCP connection on the specified port. Then, if it succeeds, it will attempt to establish an LDAP connection and do a simple search. If the search returns a result within the specified timeout and attempts, the service will be considered available. The scope of the LDAP search is limited to the immediate subordinates of the base object. The LDAP search is anonymous by default. The LDAP monitor makes use of the *com.novell.ldap.LDAPConnection* class.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.LdapMonitor |
|----------------|--|
| Remote Enabled | true |

Table 41. Monitor specific parameters for the LdapMonitor

| Paramet er | Description | Requi red | Default value |
|------------------|--|--------------|---------------------|
| dn | The distinguished name to use if authenticated search is needed. | option al | - |
| password | The password to use if authenticated search is needed. | option al | - |
| port | The destination port where connection shall be attempted. | option al | 389 |
| retry | Number of attempts to get a search result. | option al | 1 |
| searchba se | The base distinguished name to search from. | option al | base |
| searchfi lter | The LDAP search's filter. | option al | (objectcla ss=*) |

| Paramet er | Description | Requi red | Default value |
|---------------|--|--------------|------------------|
| timeout | Time in milliseconds to wait for a result from the search. | option al | 3000 |
| version | The version of the LDAP protocol to use, specified as an integer. Note: Only LDAPv3 is supported at the moment. | option al | 3 |

4.6.26. LdapsMonitor

The LDAPS monitor tests the response of an SSL-enabled LDAP server. The LDAPS monitor is an SSL-enabled extension of the LDAP monitor with a default TCP port value of 636. All LdapMonitor parameters apply, so please refer to LdapMonitor's documentation for more information.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.LdapsMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 42. Monitor specific parameters for the LdapsMonitor

| Paramete | Description | Require | Default |
|----------|--|----------|---------|
| r | | d | value |
| port | The destination port where connections shall be attempted. | optional | 636 |

Examples

4.6.27. MemcachedMonitor

This monitor allows to monitor Memcached, a distributed memory object caching system. To monitor the service availability the monitor tests if the *Memcached* statistics can be requested. The statistics are processed and stored in RRD files. The following metrics are collected:

Table 43. Collected metrics using the MemcachedMonitor

| Metric | Description |
|---------------------|--|
| uptime | Seconds the <i>Memcached</i> server has been running since last restart. |
| rusageuser | User time seconds for the server process. |
| rusagesystem | System time seconds for the server process. |
| curritems | Number of items in this servers cache. |
| totalitems | Number of items stored on this server. |
| bytes | Number of bytes currently used for caching items. |
| limitmaxbytes | Maximum configured cache size. |
| currconnections | Number of open connections to this <i>Memcached</i> . |
| totalconnections | Number of successful connect attempts to this server since start. |
| connectionstructure | Number of internal connection handles currently held by the server. |
| cmdget | Number of <i>GET</i> commands received since server startup. |
| cmdset | Number of <i>SET</i> commands received since server startup. |
| gethits | Number of successful <i>GET</i> commands (cache hits) since startup. |
| getmisses | Number of failed <i>GET</i> requests, because nothing was cached. |
| evictions | Number of objects removed from the cache to free up memory. |

| Metric | Description |
|--------------|--|
| bytesread | Number of bytes received from the network. |
| byteswritten | Number of bytes send to the network. |
| threads | Number of threads used by this server. |

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.MemcachedMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 44. Monitor specific parameters for the MemcachedMonitor

| Paramete r | Description | Require d | Default value |
|---------------|---|--------------|------------------|
| timeout | Timeout in milliseconds for Memcached connection establishment. | optional | 3000 |
| retry | Number of attempts to establish the Memcached connnection. | optional | 0 |
| port | TCP port connecting to Memcached. | optional | 11211 |

Examples

The following example shows a configuration in the poller-configuration.xml.

```
<service name="Memcached" interval="300000" user-defined="false" status="on">
    <parameter key="port" value="11211" />
    <parameter key="retry" value="2" />
    <parameter key="timeout" value="3000" />
    <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response" />
    <parameter key="ds-name" value="memcached" />
    <parameter key="rrd-base-name" value="memcached" />
    </service>

<monitor service="Memcached" class-name=
    "org.opennms.netmgt.poller.monitors.MemcachedMonitor" />
```

4.6.28. NetScalerGroupHealthMonitor

This monitor is designed for *Citrix® NetScaler®* loadbalancing checks. It checks if more than x percent of the servers assigned to a specific group on a loadbalanced service are active. The required data is gathered via SNMP from the *NetScaler®*. The status of the servers is determined by the *NetScaler®*. The provided service it self is not part of the check. The basis of this monitor is the *SnmpMonitorStrategy*. A valid SNMP configuration in OpenNMS Meridian for the *NetScaler®* is

required.



A *NetScaler*® can manage several groups of servers per application. This monitor just covers one group at a time. If there are multiple groups to check, define one monitor per group.



This monitor is not checking the loadbalanced service it self.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.NetScalerGroupHealthMonitor |
|----------------|--|
| Remote Enabled | false |

Configuration and Usage

Table 45. Monitor specific parameters for the NetScalerGroupHealthMonitor

| Parameter | Description | Requir ed | Default value |
|------------------|---|--------------|------------------|
| group-name | The name of the server group to check | require d | - |
| group- health | The percentage of active servers vs total server of the group as an integer | optional | 60 |

Examples

The following example checks a server group called *central_webfront_http*. If at least 70% of the servers are active, the service is up. If less then 70% of the servers are active the service is down. A configuration like the following can be used for the example in the poller-configuration.xml.

Details about the used SNMP checks

The monitor checks the status of the server group based on the *NS-ROOT-MIB* using the *svcGrpMemberState*. *svcGrpMemberState* is part of the *serviceGroupMemberTable*. The *serviceGroupMemberTable* is indexed by *svcGrpMemberGroupName* and *svcGrpMemberName*. A initial lookup for the group-name is performed. Based on the lookup the *serviceGroupMemberTable* is walked with the numeric representation of the server group. The monitor interprets just the server status code *7-up* as active server. Other status codes like *2-unknown* or *3-busy* are counted for total

amount of servers.

4.6.29. NrpeMonitor

This monitor allows to test plugins and checks running on the Nagios Remote Plugin Executor (NRPE) framework. The monitor allows to test the status output of any available check command executed by NRPE. Between OpenNMS Meridian and Nagios are some conceptional differences. In OpenNMS Meridian a service can only be available or not available and the response time for the service is measured. Nagios on the other hand combines service availability, performance data collection and thresholding in one check command. For this reason a Nagios check command can have more states then OK and CRITICAL. Using the NrpeMonitor marks all check command results other than OK as down. The full output of the check command output message is passed into the service down event in OpenNMS Meridian.



NRPE configuration on the server is required and the check command has to be configured, e.g. command[check_apt]=/usr/lib/nagios/plugins/check_apt



OpenNMS Meridian executes every *NRPE* check in a Java thread without *fork()* a process and it is more resource friendly. Nevertheless it is possible to run *NRPE* plugins which combine a lot of external programs like sed, awk or cut. Be aware, each command end up in forking additional processes.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.NrpeMonitor |
|----------------|--|
| Remote Enabled | false |

Table 46. Monitor specific parameters for the NrpeMonitor

| Param eter | Description | Requi red | Default value |
|---------------|---|--------------|------------------|
| retry | Number of retries before the service is marked as <i>down</i> . | option al | 0 |
| timeout | Time in milliseconds to wait for the <i>NRPE</i> executing a check command. | option al | 3000 |
| command | The {check_name} of the command configured as `command[{check_name}]="/path/to/plugin/check-script" | requir ed | empty |
| port | Port to access NRPE on the remote server. | option al | 5666 |
| padding | Padding for sending the command to the NRPE agent. | option al | 2 |

| Param | Description | Requi | Default |
|--------|---|--------------|---------|
| eter | | red | value |
| usessl | Enable encryption of network communication. NRPE uses SSL with anonymous DH and the following cipher suite TLS_DH_anon_WITH_AES_128_CBC_SHA | option al | true |

Example: Using check_apt with NRPE

This examples shows how to configure the *NrpeMonitor* running the *check_apt* command on a configured *NRPE*.

Configuration of the NRPE check command on the agent in 'nrpe.cfg'

```
command[check_apt]=/usr/lib/nagios/plugins/check_apt
```

Configuration to test the NRPE plugin with the NrpeMonitor

```
<service name="NRPE-Check-APT" interval="300000" user-defined="false" status="on">
    <parameter key="retry" value="3" />
    <parameter key="timeout" value="3000" />
    <parameter key="port" value="5666" />
    <parameter key="command" value="check_apt" />
    <parameter key="padding" value="2" />
    </service>

<monitor service="NRPE-Check-APT" class-name=
    "org.opennms.netmgt.poller.monitors.NrpeMonitor" />
```

4.6.30. NtpMonitor

The NTP monitor tests for NTP service availability. During the poll an NTP request query packet is generated. If a response is received, it is parsed and validated. If the response is a valid NTP response, the service is considered available.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.NtpMonitor |
|----------------|---|
| Remote Enabled | true |

Table 47. Monitor specific parameters for the NtpMonitor

| Paramete | Description | Require | Default |
|----------|---|----------|---------|
| r | | d | value |
| port | The destination port where the NTP request shall be sent. | optional | 123 |

| Paramete r | Description | Require d | Default value |
|---------------|--|--------------|------------------|
| retry | Number of attempts to get a response. | optional | 0 |
| timeout | Time in milliseconds to wait for a response. | optional | 5000 |

4.6.31. OmsaStorageMonitor

With OmsaStorageMonitor you are able to monitor your Dell OpenManaged servers RAID array status. The following OIDS from the STORAGEMANAGEMENT-MIB are supported by this monitor:

```
      virtualDiskRollUpStatus
      .1.3.6.1.4.1.674.10893.1.20.140.1.1.19

      arrayDiskLogicalConnectionVirtualDiskNumber
      .1.3.6.1.4.1.674.10893.1.20.140.3.1.5

      arrayDiskNexusID
      .1.3.6.1.4.1.674.10893.1.20.130.4.1.26

      arrayDiskLogicalConnectionArrayDiskNumber
      .1.3.6.1.4.1.674.10893.1.20.140.3.1.3

      arrayDiskState
      .1.3.6.1.4.1.674.10893.1.20.130.4.1.4
```

To test the status of the disk array the virtualDiskRollUpStatus is used. If the result of the virtualDiskRollUpStatus is not 3 the monitors is marked as *down*.

Table 48. Possible result of virtual disk rollup status

| Resul t | State description | Monitor state in OpenNMS Meridian |
|------------|-------------------|--------------------------------------|
| 1 | other | DOWN |
| 2 | unknown | DOWN |
| 3 | ok | UP |
| 4 | non-critical | DOWN |
| 5 | critical | DOWN |
| 6 | non-recoverable | DOWN |



You'll need to know the maximum number of possible logical disks you have in your environment. For example: If you have 3 RAID arrays, you need for each logical disk array a service poller.

To give more detailed information in case of an disk array error, the monitor tries to identify the problem using the other OIDs. This values are used to enrich the error reason in the service down event. The disk array state is resolved to a human readable value by the following status table.

Table 49. Possible array disk state errors

| Valu e | Status |
|-----------|---------------------|
| 1 | Ready |
| 2 | Failed |
| 3 | Online |
| 4 | Offline |
| 6 | Degraded |
| 7 | Recovering |
| 11 | Removed |
| 15 | Resynching |
| 24 | Rebuilding |
| 25 | noMedia |
| 26 | Formating |
| 28 | Running Diagnostics |
| 35 | Initializing |

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.OmsaStorageMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

Monitor specific parameters for the OmsaStorageMonitor

| Parameter | Description | _ | Default value |
|-----------------------|--|--------------|--------------------------|
| virtualDisk Number | The disk index of your RAID array | option al | 1 |
| retry | Amount of attempts opening a connection and try to get the greeting banner before the service goes down. | | from snmp- config.xml |

| Parameter | Description | _ | Default value |
|-----------|--|--------------|--------------------------|
| timeout | Time in milliseconds to wait before receiving the SNMP response. | option al | from snmp- config.xml |
| port | The TCP port OpenManage is listening | option al | from snmp- config.xml |

Some example configuration how to configure the monitor in the poller-configuration.xml.

The RAID array monitor for your first array is configured with virtualDiskNumber = 1 and can look like this:

```
<service name="OMSA-Disk-Array-1" interval="300000" user-defined="false" status="on">
    <parameter key="retry" value="3"/>
    <parameter key="timeout" value="6000"/>
    <parameter key="virtualDiskNumber" value="1"/>
    </service>

<monitor service="OMSA-Disk-Array-1" class-name=
    "org.opennms.netmgt.poller.monitors.OmsaStorageMonitor"/>
```

If there is more than one RAID array to monitor you need an additional configuration. In this case virtualDiskNumber = 2. And so on...

```
<service name="OMSA-Disk-Array-2" interval="300000" user-defined="false" status="on">
    <parameter key="retry" value="3"/>
    <parameter key="timeout" value="6000"/>
    <parameter key="virtualDiskNumber" value="2"/>
    </service>

<monitor service="OMSA-Disk-Array-2" class-name=
    "org.opennms.netmgt.poller.monitors.OmsaStorageMonitor"/>
```

4.6.32. OpenManageChassisMonitor

The OpenManageChassis monitor tests the status of a Dell chassis by querying its SNMP agent. The monitor polls the value of the node's SNMP OID .1.3.6.1.4.1.674.10892.1.300.10.1.4.1 (MIB-Dell-10892::chassisStatus). If the value is OK (3), the service is considered available.

As this monitor uses SNMP, the queried nodes must have proper SNMP configuration in *snmp-config.xml*.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.OpenManageChassisMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

Table 50. Monitor specific parameters for the OpenManageChassisMonitor

| Paramet er | Description | Requir ed | Default value |
|---------------|--|--------------|----------------------------------|
| port | The port to which connection shall be tried. | optional | from snmp- config.xml |
| retry | Number of polls to attempt. | optional | <pre>from snmp- config.xml</pre> |
| timeout | Time (in milliseconds) to wait before receiving the SNMP response. | optional | from snmp- config.xml |

Examples

Dell MIBs

Dell MIBs can be found here. Download the DCMIB<version>.zip or DCMIB<version>.exe file corresponding to the version of your OpenManage agents. The latest one should be good enough for all previous version though.

4.6.33. PageSequenceMonitor

The *PageSequenceMonitor* (PSM) allows OpenNMS to monitor web applications. This monitor has several configuration options regarding *IPv4*, *IPv6* and how to deal with name resolution. To add flexibility, the node label and IP address can be passed as variable into the monitor. This allows running the monitor with node dependent configuration. Beyond testing a web application with a single *URL* it can also test a path through a web application. A test path through an web application can look like this:

- 1. login to a certain web application
- 2. Execute an action while being logged in
- 3. Log off

The service is considered as *up* if all this is working ok. If there's an error somewhere, your application will need attention and the service changes the state to *down*.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.PageSequenceMonitor |
|----------------|--|
| Remote Enabled | true |

Configuration and Usage

The configuration for this monitor consists of several parts. First is the overall configuration for retries and timeouts. These parameters are global for the whole path through the web application.

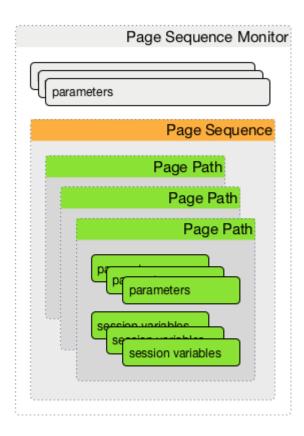


Figure 16. Configuration overview of the PSM

The overall layout of the monitor configuration is more complex. Additionally, it is possible to configure a page sequence containing a path through a web application.

Table 51. Monitor parameters for the PageSequenceMonitor

| Paramete r | Description | Requi red | Default value |
|---------------|---|--------------|------------------|
| timeout | Time in milliseconds to wait before assuming that a packet has not responded. | option al | 3000 |
| retry | The number of retries per page. | option al | 0 |

| Paramete r | Description | Requi red | Default value |
|--------------------|---|--------------|------------------|
| strict- timeout | Defines a timer to wait before a retry attempt is made. It is only used if at least one (1) retry is configured. If retry >= 1 and strict-timeout is true the next attempt is delayed and the Poller Daemon waits NOW - InitialAttempt ms + Timeout ms. With strict-timout = false the next attempt is started right after a failure. | option al | false |
| page- sequence | Definition of the page-sequence to execute, see table with <i>Page Sequence Parameter</i> | requir ed | - |
| sequence- retry | The retry parameter for the entire page sequence. | option al | 0 |

Table 52. Page Sequence Parameter

| Parameter | Description | Requ ired | Default |
|--------------|--|--------------|--|
| name | The name of the page-sequence. (Is this relevant/used?) | optio nal | - |
| method | HTTP method for example GET or POST | - | - |
| http-version | HTTP protocol version number, 0.9, 1.0 or 1.1 | optio nal | HTTP/1.1 |
| user-agent | Set the <i>user agent</i> field in <i>HTTP</i> header to identify the OpenNMS monitor | optio nal | OpenNMS PageSequenceMonitor (Service name: "\${SERVICE NAME}") |
| virtual-host | Set the <i>virtual host</i> field in <i>HTTP</i> header. In case of an HTTPS request, this is also the virtual domain to send as part of the <i>TLS</i> negotiation, known as server name indication (<i>SNI</i>) (See: RFC3546 section 3.1) | - | |
| path | The relative URL to call in the request. | requi red | - |
| scheme | Define the <i>URL</i> scheme as http or https | optio nal | http |
| user-info | Set user info field in the HTTP header | - | - |
| host | Set <i>host</i> field in <i>HTTP</i> header | optio nal | IP interface address of the service |
| requireIPv6 | Communication requires a connection to an <i>IPv6</i> address. (true or false) | - | - |
| requireIPv4 | Communication requires a connection to an <i>IPv4</i> address. (true or false) | - | - |

| Parameter | Description | Requ ired | Default |
|------------------------------|---|--------------|---------|
| disable-ssl- verification | Enable or disable SSL certificate verification for HTTPS tests. Please use this option carefully, for self-signed certificates import the CA certificate in the JVM and don't just disable it. | optio nal | false |
| port | Port of the web server connecting to | optio nal | 80 |
| query | ?? | _ | - |
| failureMatch | Text to look for in the response body. This is a <i>Regular Expression</i> matched against every line, and it will be considered a failure at the first match and sets the service with this monitor <i>Down</i> . | - | |
| failureMessa ge | The failure message is used to construct the reason code. \$\{n}\ values may be used to pull information from matching groups in the failureMatch regular expression. | - | |
| successMatch | Text to look for in the response body. This is a <i>Regular Expression</i> matched against every line, and it will be considered a success at the first match and sets the service with this monitor <i>Up</i> . | optio nal | _ |
| locationMatc h | The relative URL which must be loaded for the request to be considered successful. | optio nal | - |
| response- range | Range for allowed HTTP error codes from the response. | _ | - |
| session- variable | Assign the value of a regex match group to a session variable with a user-defined name. The match group is identified by number and must be zero or greater. | - | - |
| response- range | A comma-separated list of acceptable <i>HTTP</i> response code ranges (200-202,299). | optio nal | 100-399 |



If you set requireIPv4 and requireIPv6 false, the host IP for connection will be resolved from system name resolver and the associated IP address from the IP interface is ignored.

Table 53. Variables which can be passed in the configuration

| Variable | Description |
|---------------|---|
| \${nodelabel} | Nodelabel of the node the monitor is associated to. |

Session variables

It is possible to assign strings from a retrieved page to variables that can be used in page parameters later in the same sequence. First, specify one or more capturing groups in the successMatch expression (see Java Class Pattern for more information on regular expressions in Java). The captured values can then be assigned to variable names by using the session-variable parameter, and used in a later page load.

Per-page response times

It is possible to collect response times for individual pages in a sequence. To use this functionality, a ds-name attribute must be added to each page whose load time should be tracked. The response time for each page will be stored in the same *RRD* file specified for the service via the rrd-base-name parameter under the specified datasource name.



You will need to delete existing *RRD* files and let them be recreated with the new list of datasources when you add a ds-name attribute to a page in a sequence that is already storing response time data.

Examples

The following example shows how to monitor the *OpenNMS* web application using several mechanisms. It first does an *HTTP GET* of \${ipaddr}/opennms (following redirects as a browser would) and then checks to ensure that the resulting page has the phrase Password on it. Next, a login is attempted using *HTTP POST* to the relative *URL* for submitting form data (usually, the *URL* which the form action points to). The parameters (j_username and j_password) indicate the form's data and values to be submitted. After getting the resulting page, first the expression specified in the page's failureMatch attribute is verified, which when found anywhere on the page indicates that the page has failed. If the failureMatch expression is not found in the resulting page, then the expression specified in the page's successMatch attribute is checked to ensure it matches the resulting page. If the successMatch expression is not found on the page, then the page fails. If the monitor was able to successfully login, then the next page is processed. In the example, the monitor navigates to the Event page, to ensure that the text *Event Queries* is found on the page. Finally, the monitor calls the URL of the logout page to close the session. By using the locationMatch parameter, it is verified that the logout was successful and a redirect was triggered.



Each page is checked to ensure its *HTTP* response code fits into the responserange, before the failureMatch, successMatch, and locationMatch expressions are evaluated.

```
<service name="OpenNMS-Web-Login" interval="30000" user-defined="true" status="on">
  <parameter key="retry" value="1"/>
 <parameter key="timeout" value="5000"/>
 <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
 <parameter key="ds-name" value="opennmslogin"/>
 <parameter key="page-sequence">
    <page-sequence>
      <page path="/opennms/login.jsp"</pre>
            port="8980"
            successMatch="Password" />
      <page path="/opennms/j_spring_security_check"</pre>
            port="8980"
            method="POST">
        <parameter key="j_username" value="admin"/>
        <parameter key="j_password" value="admin"/>
      </page>
      <page path="/opennms/index.jsp"</pre>
            port="8980"
            successMatch="Log Out" />
      <page path="/opennms/event/index"</pre>
            port="8980" successMatch="Event Queries" />
      <page path="/opennms/j_spring_security_logout"</pre>
            port="8980"
            method="POST"
            response-range="300-399"
            locationMatch="/opennms" />
    </page-sequence>
 </parameter>
</service>
<monitor service="OpenNMS-Web-Login" class-name=</pre>
"org.opennms.netmgt.poller.monitors.PageSequenceMonitor"/>
```

```
<service name="OpenNMS-Web-Login" interval="30000" user-defined="true" status="on">
 <parameter key="retry" value="1"/>
 <parameter key="timeout" value="5000"/>
 <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
 <parameter key="ds-name" value="opennmslogin"/>
 <parameter key="page-sequence">
    <page-sequence>
      <page scheme="http"</pre>
            host="ecomm.example.com"
            port="80"
            path="/ecomm/jsp/Login.jsp"
            virtual-host="ecomm.example.com"
            successMatch="eComm Login"
            timeout="10000"
            http-version="1.1"/>
      <page scheme="https"</pre>
            method="POST"
            host="ecomm.example.com" port="443"
            path="/ecomm/controller"
            virtual-host="ecomm.example.com"
            successMatch="requesttab select.gif"
            failureMessage="Login failed: ${1}"
            timeout="10000"
            http-version="1.1">
        <parameter key="action_name" value="XbtnLogin"/>
        <parameter key="session_timeout" value=""/>
        <parameter key="userid" value="EXAMPLE"/>
        <parameter key="password" value="econ"/>
      </page>
      <page scheme="http"</pre>
            host="ecomm.example.com" port="80"
            path="/econsult/controller"
            virtual-host="ecomm.example.com"
            successMatch="You have successfully logged out of eComm"
            timeout="10000" http-version="1.1">
        <parameter key="action_name" value="XbtnLogout"/>
      </page>
    </page-sequence>
 </parameter>
</service>
<monitor service="OpenNMS-Web-Login" class-name=</pre>
"org.opennms.netmgt.poller.monitors.PageSequenceMonitor"/>
```

```
<service name="OpenNMS-Web-Login" interval="30000" user-defined="true" status="on">
 <parameter key="retry" value="1"/>
 <parameter key="timeout" value="5000"/>
 <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
 <parameter key="ds-name" value="opennmslogin"/>
 <parameter key="page-sequence">
    <page-sequence name="opennms-login-seq-dynamic-credentials">
      <page path="/opennms"</pre>
            port="80"
            virtual-host="demo.opennms.org"
            successMatch="(?s)User:.*<strong>(.*?)</strong>.*?Password:.*?
<strong>(.*?)</strong>">
        <session-variable name="username" match-group="1" />
        <session-variable name="password" match-group="2" />
      </page>
      <page path="/opennms/j_acegi_security_check"</pre>
            port="80"
            virtual-host="demo.opennms.org"
            method="POST"
            failureMatch="(?s)Your log-in attempt failed.*Reason: ([^<]*)"</pre>
            failureMessage="Login Failed: ${1}"
            successMatch="Log out">"
        <parameter key="j_username" value="${username}" />
        <parameter key="j_password" value="${password}" />
      </page>
      <page path="/opennms/event/index.jsp"</pre>
            port="80"
            virtual-host="demo.opennms.org"
            successMatch="Event Queries" />
      <page path="/opennms/j acegi logout"</pre>
            port="80"
            virtual-host="demo.opennms.org"
            successMatch="logged off" />
    </page-sequence>
 </parameter>
</service>
<monitor service="OpenNMS-Web-Login" class-name=</pre>
"org.opennms.netmgt.poller.monitors.PageSequenceMonitor"/>
```

```
<service name="OpenNMS-Demo-Login" interval="300000" user-defined="true" status="on">
 <parameter key="page-sequence">
    <page-sequence>
      <page path="/opennms"</pre>
            port="80"
            virtual-host="demo.opennms.org"
            successMatch="(?s)User:.*<strong>(.*?)</strong>.*?Password:.*?
<strong>(.*?)</strong>">
        <session-variable name="username" match-group="1" />
        <session-variable name="password" match-group="2" />
      </page>
      <page path="/opennms/j_acegi_security_check"</pre>
            port="80"
            virtual-host="demo.opennms.org"
            method="POST"
            successMatch="Log out">"
        <parameter key="j_username" value="${username}" />
        <parameter key="j_password" value="${password}" />
      </page>
      <page path="/opennms/j_acegi_logout"</pre>
            port="80"
            virtual-host="demo.opennms.org"
            successMatch="logged off" />
    </page-sequence>
 </parameter>
</service>
<monitor service="OpenNMS-Demo-Login" class-name=</pre>
"org.opennms.netmgt.poller.monitors.PageSequenceMonitor"/>
```

```
<service name="OpenNMS-Login" interval="300000" user-defined="false" status="on">
 <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
  <parameter key="rrd-base-name" value="opennmslogin"/>
  <parameter key="ds-name" value="overall"/>
  <parameter key="page-sequence">
    <page-sequence>
      <page path="/opennms/acegilogin.jsp"</pre>
            port="8980"
            ds-name="login-page"/>
      <page path="/opennms/event/index.jsp"</pre>
            port="8980"
            ds-name="event-page"/>
    </page-sequence>
 </parameter>
</service>
<monitor service="OpenNMS-Login" class-name=</pre>
"org.opennms.netmgt.poller.monitors.PageSequenceMonitor"/>
```

4.6.34. PercMonitor

This monitor tests the status of a PERC RAID array.

The monitor first polls the RAID-Adapter-MIB::logicaldriveTable (1.3.6.1.4.1.3582.1.1.2) to retrieve the status of the RAID array you want to monitor. If the value of the status object of the corresponding logicaldriveEntry is not 2, the array is degraded and the monitor further polls the RAID-Adapter-MIB::physicaldriveTable (1.3.6.1.4.1.3582.1.1.3) to detect the failed drive(s).



This monitor requires the outdated persnmpd software to be installed on the polled nodes. Please prefer using OmsaStorageMonitor monitor where possible.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.PercMonitor | | |
|----------------|--|--|--|
| Remote Enabled | false (relies on SNMP configuration) | | |

Table 54. Monitor specific parameters for the PercMonitor

| Param eter | Description | Requi red | Default value |
|---------------|-------------------------------------|--------------|---------------|
| array | The RAID array you want to monitor. | option al | 0.0 |

| Param eter | Description | Requi red | Default value |
|---------------|---|--------------|--------------------------|
| port | The UDP port to connect to | option al | from snmp- config.xml |
| retry | The number of attempts the monitor shall try getting a response. | option al | from snmp- config.xml |
| timeout | The amount of thime in milliseconds the monitor shall wait for a reponse during each polling attempt. | option al | from snmp- config.xml |

```
<!-- Monitor 1st RAID arrays using configuration from snmp-config.xml -->
<service name="PERC" interval="300000" user-defined="false" status="on" />
<monitor service="PERC" class-name="org.opennms.netmgt.poller.monitors.PercMonitor" />
```

4.6.35. Pop3Monitor

The POP3 monitor tests for POP3 service availability on a node. The monitor first tries to establish a TCP connection on the specified port. If a connection is established, a service banner should have been received. The monitor makes sure the service banner is a valid POP3 banner (ie: starts with "+OK"). If the banner is valid, the monitor sends a *QUIT* POP3 command and makes sure the service answers with a valid response (ie: a response that starts with "+OK"). The service is considered available if the service's answer to the *QUIT* command is valid.

The behaviour can be simulated with telnet:

```
$ telnet mail.opennms.org 110
Trying 192.168.0.100
Connected to mail.opennms.org.
Escape character is '^]'.
+OK <21860.1076718099@mail.opennms.org>
quit
+OK
Connection closed by foreign host.
```

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.Pop3Monitor | | |
|----------------|--|--|--|
| Remote Enabled | true | | |

Configuration and Usage

Table 55. Monitor specific parameters for the Pop3Monitor

| Paramete r | Description | Requi red | Default value |
|--------------------|---|--------------|------------------|
| port | TCP port to connect to. | option al | 110 |
| retry | Number of attempts to find the service available. | option al | 0 |
| strict- timeout | Boolean If set to true, makes sure that at least timeout milliseconds are elapsed between attempts. | option al | false |
| timeout | Timeout in milliseconds for the underlying socket's <i>connect</i> and <i>read</i> operations. | option al | 3000 |

4.6.36. PrTableMonitor

The PrTableMonitor monitor tests the prTable of a net-snmp SNMP agent.

A table containing information on running programs/daemons configured for monitoring in the snmpd.conf file of the agent. Processes violating the number of running processes required by the agent's configuration file are flagged with numerical and textual errors.

```
— UCD-SNMP-MIB
```

The monitor looks up the *prErrorFlag* entries of this table. If the value of a *prErrorFlag* entry in this table is set to "1" the service is considered unavailable.

A Error flag to indicate trouble with a process. It goes to 1 if there is an error, 0 if no error.

```
— UCD-SNMP-MIB
```

Monitor facts

| Remote Enabled | false |
|----------------|-------|
|----------------|-------|

Configuration and Usage

Table 56. Monitor specific parameters for the PrTableMonitor

| Param eter | Description | Requi red | Default value |
|---------------|---|--------------|--------------------------|
| port | The port to which connection shall be tried. | option al | from snmp- config.xml |
| retry | Number of polls to attempt. | option al | from snmp- config.xml |
| retries | Deprecated. Same as retry. Parameter retry takes precedence if both are set. | option al | from snmp- config.xml |
| timeout | Time in milliseconds to wait before receiving the SNMP response. | option al | from snmp- config.xml |

Examples

UCD-SNMP-MIB

The UCD-SNMP-MIB may be found here.

4.6.37. RadiusAuthMonitor

This monitor allows to test the functionality of the RADIUS authentication system. The availability is tested by sending an *AUTH* packet to the *RADIUS server*. If a valid *ACCEPT* response is received, the *RADIUS* service is *up* and considered as available.



To use this monitor it is required to install the *RADIUS* protocol for OpenNMS Meridian.

```
{apt-get,yum} install {opennms-package-base-name}-plugin-protocol-radius
```

The test is similar to test the behavior of a *RADIUS* server by evaluating the result with the command line tool radtest.

1 The Access-Accept message which is evaluated by the monitor.

Monitor facts

| Class Name | org.opennms.protocols.radius.monitor.RadiusAuthMonitor |
|----------------|--|
| Remote Enabled | false |

Table 57. Monitor specific parameters for the RadiusAuthMonitor

| Param eter | Description | Requi red | Default value |
|---------------|--|--------------|------------------|
| timeout | Time in milliseconds to wait for the <i>RADIUS</i> service. | option al | 5000 |
| retry | This is a placeholder for the second optional monitor parameter description. | option al | 0 |
| authpor t | RADIUS authentication port. | option al | 1812 |
| acctpor t | RADIUS accounting port. | | 1813 |
| user | Username to test the authentication | | OpenNMS |
| passwor d | Password to test the authentication | | OpenNMS |
| secret | The <i>RADIUS</i> shared secret used for communication between the <i>client/NAS</i> and the <i>RADIUS</i> server. | | secret |
| authtyp e | RADIUS authentication type. The following authentication types are supported: chap, pap, mschapv1, mschapv2, eapmd5, eapmschapv2 | | pap |
| nasid | The Network Access Server identifier originating the Access-Request. | option al | opennms |

Example configuration how to configure the monitor in the poller-configuration.xml.

4.6.38. SmbMonitor

This monitor is used to test the *NetBIOS over TCP/IP* name resolution in Microsoft Windows environments. The monitor tries to retrieve a *NetBIOS name* for the IP address of the interface. Name services for *NetBIOS* in Microsoft Windows are provided on port 137/UDP or 137/TCP.

The service uses the IP address of the interface, where the monitor is assigned to. The service is *up* if for the given IP address a *NetBIOS name* is registered and can be resolved.

For troubleshooting see the usage of the Microsoft Windows command line tool nbtstat or on Linux nmblookup.



Microsoft deprecated the usage of *NetBIOS*. is used as the default name resolution.

Since Windows Server 2000 DNS

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.SmbMonitor |
|----------------|---|
| Remote Enabled | false |

Table 58. Monitor specific parameters for the SmbMonitor

| Parameter | Description | Require d | Default value |
|-----------|--|--------------|------------------|
| retry | Number of attempts to get a valid response | required | - |
| timeout | Timeout in milliseconds for TCP connection establishment | required | - |

| Parameter | Description | Require d | Default value |
|--------------------|---|--------------|------------------|
| do-node- status | Try to get the NetBIOS node status type for the given address | optional | true |

Some example configuration how to configure the monitor in the poller-configuration.xml.

```
<service name="SMB" interval="300000" user-defined="false" status="on">
   <parameter key="retry" value="1"/>
   <parameter key="timeout" value="3000"/>
   </service>
</monitor service="SMB" class-name="org.opennms.netmgt.poller.monitors.SmbMonitor"/>
```

4.6.39. SnmpMonitor

The SNMP monitor gives a generic possibility to monitor states and results from SNMP agents. This monitor has two basic operation modes:

- Test the response value of one specific *OID* (scalar object identifier);
- Test multiple values in a whole *table*.

To decide which mode should be used, the walk and match-all parameters are used.

See the Operating mode selection'' and Monitor specific parameters for the SnmpMonitor' tables below for more information about these operation modes.

Table 59. Operating mode selection

| wal k | match- | Operating mode |
|-----------|--------|---|
| true | true | tabular, all values must match |
| | false | tabular, any value must match |
| | count | specifies that the value of at least minimum and at most maximum objects encountered in |
| fals e | true | scalar |
| C | false | scalar |
| | count | tabular, between minimum and maximum values must match |



This monitor can't be used on the OpenNMS Meridian Remote Poller. It is currently not possible for the Remote Poller to have access to the SNMP configuration of a central OpenNMS Meridian.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.SnmpMonitor |
|----------------|--|
| Remote Enabled | false |

When the monitor is configured to persist the response time, it will count the total amount of time spent until a successful response is obtained, including the retries. It won't store the time spent during the last successful attempt.

Table 60. Monitor specific parameters for the SnmpMonitor

| Paramet er | Description | Requ ired | Default value |
|---------------|--|--------------|---------------|
| hex | Specifies that the value monitored should be compared against its hexadecimal representation. Useful when the monitored value is a string containing non-printable characters. | optio nal | false |
| match-all | Can be set to: count: specifies that the value of at least minimum and at most maximum objects encountered in the walk must match the criteria specified by operand and operator. true and walk is set to true: specifies that the value of every object encountered in the walk must match the criteria specified by the operand and operator parameters. false and walk is set to true: specifies that the value of any object encountered in the walk must match the criteria specified by the operand and operator parameters. | optio nal | true |
| maximum | Valid only when match-all is set to count, otherwise ignored. Should be used in conjunction with the minimum parameter. Specifies that the value of at most maximum objects encountered in the walk must meet the criteria specified by the operand and operator parameters. | optio nal | 0 |
| minimum | Valid only when match-all is set to count, otherwise ignored. Should be used in conjunction with the maximum parameter. Specifies that the value of at least minimum objects encountered in the walk must meet the criteria specified by the operand and operator parameters. | optio nal | 0 |

| Paramet er | Description | Requ ired | Default value |
|---------------------|---|--------------|---|
| oid | The object identifier of the <i>MIB</i> object to monitor. If no other parameters are present, the monitor asserts that the agent's response for this object must include a valid value (as opposed to an error, no-such-name, or end-of-view condition) that is non-null. | optio nal | .1.3.6.1.2.1.1.2.0 (SNMPv2- MIB::SysObjectID) |
| operand | The value to be compared against the observed value of the monitored object. Comparison will always succeed if either the operand or operator parameter isn't set and the monitored value is non-null. | optio nal | - |
| operator | The operator to be used for comparing the monitored object against the operand parameter. Must be one of the following symbolic operators: 81t; (<): Less than. Both operand and observed object value must be numeric. 8gt; (>): Greater than. Both operand and observed object value must be numeric. 81t;= (<): Less than or equal to. Both operand and observed object value must be numeric. 8gt;= (>=): Greater than or equal to. Both operand and observed object value must be numeric. =: Equal to. Applied in numeric context if both operand and observed object value are numeric, otherwise in string context as a case-sensitive exact match. !=: Not equal to. Applied in numeric context if both operand and observed object value are numeric, otherwise in string context as a case-sensitive exact match. *: Regular expression match. Always applied in string context. Note: Comparison will always succeed if either the operand or operator parameter isn't set and the monitored value is non-null. Keep in mind that you need to escape all < and > characters as XML entities (81t; and 8gt; respectively) | optio nal | |
| port | Destination port where the SNMP requests shall be sent. | optio nal | from snmp-config.xml |
| reason- template | A user-provided template used for the monitor's reason code if the service is unvailable. Defaults to a reasonable value if unset. See below for an explanation of the possible template parameters. | optio nal | depends on operation mode |

| Paramet er | Description | Requ ired | Default value |
|---------------|--|--------------|----------------------|
| retry | Number of polls to attempt. | optio nal | from snmp-config.xml |
| retries | Deprecated Same as retry. Parameter retry takes precedence if both are set. | optio nal | from snmp-config.xml |
| timeout | Timeout in milliseconds for retrieving the object's value. | optio nal | from snmp-config.xml |
| walk | false: Sets the monitor to poll for a scalar object unless if the match-all parameter is set to count, in which case the match-all parameter takes precedence. true: Sets the monitor to poll for a tabular object where the match-all parameter defines how the tabular object's values must match the criteria defined by the operator and operand parameters. See also the match-all, minimum, and maximum parameters. | optio nal | false |

Table 61. Variables which can be used in the reason-template parameter

| Variable | Description |
|-------------------------------|--|
| \${hex} | Value of the hex parameter. |
| \${ipaddr} | IP address polled. |
| \${matchAll} | Value of the match-all parameter. |
| \${matchCount} | When match-all is set to count, contains the number of matching instances encountered. |
| \${maximum} | Value of the maximum parameter. |
| \${minimum} | Value of the minimum paramater. |
| <pre>\${observedValue }</pre> | Polled value that made the monitor succeed or fail. |
| \$ {oid} | Value of the oid parameter. |
| \${operand} | Value of the operand parameter. |
| <pre>\${operator}</pre> | Value of the operator parameter. |
| \${port} | Value of the port parameter. |
| \${retry} | Value of the retry parameter. |
| \${timeout} | Value of the timeout parameter. |
| \${walk} | Value of the walk parameter. |

Example for monitoring scalar object

As a working example we want to monitor the thermal system fan status which is provided as a scalar object ID.

```
cpqHeThermalSystemFanStatus .1.3.6.1.4.1.232.6.2.6.4.0
```

The manufacturer *MIB* gives the following information:

Description of the cpqHeThermalSystemFanStatus from CPQHLTH-MIB

```
SYNTAX INTEGER {
    other
             (1),
    ok
             (2),
    degraded (3),
    failed
            (4)
}
ACCESS read-only
DESCRIPTION
"The status of the fan(s) in the system.
This value will be one of the following:
other(1)
Fan status detection is not supported by this system or driver.
ok(2)
All fans are operating properly.
degraded(3)
A non-required fan is not operating properly.
failed(4)
A required fan is not operating properly.
If the cpqHeThermalDegradedAction is set to shutdown(3) the
system will be shutdown if the failed(4) condition occurs."
```

The SnmpMonitor is configured to test if the fan status returns ok(2). If so, the service is marked as up. Any other value indicates a problem with the thermal fan status and marks the service down.

Example SnmpMonitor as HP InsightManager fan monitor in poller-configuration.xml

- ① Scalar object ID to test
- 2 Operator for testing the response value
- 3 Integer 2 as operand for the test
- 4 Encode MIB status in the reason code to give more detailed information if the service goes down

Example test SNMP table with all matching values

The second mode shows how to monitor values of a whole SNMP table. As a practical use case the status of a set of physical drives is monitored. This example configuration shows the status monitoring from the CPQIDA-MIB.

We use as a scalar object id the physical drive status given by the following tabular OID:

```
cpqDaPhyDrvStatus .1.3.6.1.4.1.232.3.2.5.1.1.6
```

```
SYNTAX INTEGER {
    other
                      (1),
    ok
                      (2),
    failed
                      (3),
    predictiveFailure (4)
}
ACCESS read-only
DESCRIPTION
Physical Drive Status.
This shows the status of the physical drive.
The following values are valid for the physical drive status:
other (1)
 Indicates that the instrument agent does not recognize
 the drive. You may need to upgrade your instrument agent
 and/or driver software.
ok (2)
 Indicates the drive is functioning properly.
failed (3)
 Indicates that the drive is no longer operating and
 should be replaced.
predictiveFailure(4)
 Indicates that the drive has a predictive failure error and
 should be replaced.
```

The configuration in our monitor will test all physical drives for status ok(2).

Example SnmpMonitor as HP Insight physical drive monitor in poller-configuration.xml

- 1 OID for SNMP table with all physical drive states
- ② Enable walk mode to test every entry in the table against the test criteria
- 3 Test operator for integer
- 4 Integer 2 as operand for the test
- 5 Test in walk mode has to be passed for every entry in the table
- 6 Encode MIB status in the reason code to give more detailed information if the service goes down

Example test SNMP table with all matching values

This example shows how to use the SnmpMonitor to test if the number of static routes are within a given boundary. The service is marked as *up* if at least 3 and at maxium 10 static routes are set on a network device. This status can be monitored by polling the table *ipRouteProto* from the RFC1213-MIB2.

```
ipRouteProto 1.3.6.1.2.1.4.21.1.9
```

The MIB description gives us the following information:

```
SYNTAX INTEGER {
    other(1),
    local(2),
    netmgmt(3),
    icmp(4),
    egp(5),
    ggp(6),
    hello(7),
    rip(8),
    is-is(9),
    es-is(10),
    ciscoIgrp(11),
    bbnSpfIgp(12),
    ospf(13),
    bgp(14)}
}
ACCESS read-only
DESCRIPTION
"The routing mechanism via which this route was learned.
Inclusion of values for gateway routing protocols is not
intended to imply that hosts should support those protocols."
```

To monitor only local routes, the test should be applied only on entries in the *ipRouteProto* table with value 2. The number of entries in the whole *ipRouteProto* table has to be counted and the boundaries on the number has to be applied.

Example SnmpMonitor used to test if the number of local static route entries are between 3 or 10.

```
<service name="All-Static-Routes" interval="300000" user-defined="false" status="on">
    <parameter key="oid" value=".1.3.6.1.2.1.4.21.1.9" /> ①
    <parameter key="walk" value="true" /> ②
    <parameter key="operator" value="=" /> ③
    <parameter key="operand" value="2" /> ④
    <parameter key="match-all" value="count" /> ⑤
    <parameter key="minimum" value="3" /> ⑥
    <parameter key="maximum" value="10" /> ⑦
    </service>

<monitor service="All-Static-Routes" class-name=
"org.opennms.netmgt.poller.monitors.SnmpMonitor" />
```

- ① OID for SNMP table *ipRouteProto*
- ② Enable walk mode to test every entry in the table against the test criteria
- 3 Test operator for integer
- 4 Integer 2 as operand for testing local route entries
- (5) Test in *walk mode* has is set to count to get the number of entries in the table regarding operator and operand
- 6 Lower count boundary set to 3
- Tigh count boundary is set to 10

4.6.40. SshMonitor

The SSH monitor tests the availability of a SSH service. During the poll an attempt is made to connect on the specified port. If the connection request is successful, then the service is considered up. Optionaly, the banner line generated by the service may be parsed and compared against a pattern before the service is considered up.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.SshMonitor |
|----------------|---|
| Remote Enabled | true |

Table 62. Monitor specific parameters for the SshMonitor

| Paramet er | Description | Requi red | Default value |
|---------------|--|--------------|---------------|
| banner | Regular expression to be matched against the service's banner. | option al | - |

| Paramet er | Description | Requi red | Default value |
|-------------------|--|--------------|--------------------------|
| client- banner | The client banner that OpenNMS Meridian will use to identify itself on the service. | option al | SSH-1.99- OpenNMS_1.5 |
| match | Regular expression to be matched against the service's banner. Deprecated, please use the banner parameter instead. Note that this parameter takes precedence over the banner parameter, though. | option al | - |
| port | TCP port to which SSH connection shall be tried. | option al | 22 |
| retry | Number of attempts to establish the SSH connnection. | option al | 0 |
| timeout | Timeout in milliseconds for SSH connection establishment. | option al | 3000 |

```
<service name="SSH" interval="300000" user-defined="false" status="on">
    <parameter key="retry" value="1"/>
    <parameter key="banner" value="SSH"/>
    <parameter key="client-banner" value="OpenNMS poller"/>
    <parameter key="timeout" value="5000"/>
    <parameter key="rrd-repository" value="/var/lib/opennms/rrd/response"/>
    <parameter key="rrd-base-name" value="ssh"/>
    <parameter key="ds-name" value="ssh"/>
    </service>
<monitor service="SSH" class-name="org.opennms.netmgt.poller.monitors.SshMonitor"/>
```

4.6.41. SSLCertMonitor

This monitor is used to test if a SSL certificate presented by a remote network server are valid. A certificate is invalid if its initial time is prior to the current time, or if the current time is prior to 7 days (configurable) before the expiration time. The monitor only supports SSL on the socket and does not support a higher level protocol above it.

You can simulate the behavior by running a command like this:

```
echo | openssl s_client -connect <site>:<port> 2>/dev/null | openssl x509 -noout -dates
```

The output shows you the time range a certificate is valid:

```
notBefore=Dec 24 14:11:34 2013 GMT
notAfter=Dec 25 10:37:40 2014 GMT
```

You can configure a threshold in days applied on the notAfter date.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.SSLCertMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 63. Monitor specific parameters for the SSLCertMonitor

| Parame ter | Description | Requi red | Default value |
|-----------------|---|--------------|------------------|
| port | TCP port for the service with SSL certificate. | requir ed | -1 |
| retry | Number of attempts to get the certificate state | option al | 0 |
| timeout | Time in milliseconds to wait before next attempt. | option al | 3000 |
| days | Number of days before the certificate expires that we mark the service as failed. | option al | 7 |
| server- name | This is the DNS hostname to send as part of the <i>TLS</i> negotiation, known as server name indication (<i>SNI</i>) (See: RFC3546 section 3.1) | option al | |

Table 64. Variables which can be passed in the configuration for server-name

| Variable | Description |
|---------------|---|
| \${ipaddr} | The node's IP-Address |
| \${nodeid} | The node ID |
| \${nodelabel} | Label of the node the monitor is associated to. |
| \${svcname} | The service name |



The monitor has no support for communicating on other protocol layers above the SSL session layer. It is not able to send a Host header for HTTPS, or issue a STARTTLS command for IMAP, POP3, SMTP, FTP, XMPP, LDAP, or NNTP.

Examples

The following example shows how to monitor SSL certificates on services like IMAPS, SMTPS and HTTPS. If the certificates expire within 30 days the service goes down and indicates this issue in the reason of the monitor. In this example the monitoring interval is reduced to test the certificate every 2 hours (7,200,000 ms). Configuration in poller-configuration.xml is as the following:

```
<service name="SSL-Cert-IMAPS-993" interval="7200000" user-defined="false" status="on</pre>
    <parameter key="retry" value="2"/>
    <parameter key="timeout" value="2000"/>
    <parameter key="port" value="993"/>
    <parameter key="days" value="30"/>
</service>
<service name="SSL-Cert-SMTPS-465" interval="7200000" user-defined="false" status="on</pre>
    <parameter key="retry" value="2"/>
    <parameter key="timeout" value="2000"/>
    <parameter key="port" value="465"/>
    <parameter key="days" value="30"/>
</service>
<service name="SSL-Cert-HTTPS-443" interval="7200000" user-defined="false" status="on</pre>
    <parameter key="retry" value="2"/>
    <parameter key="timeout" value="3000"/>
    <parameter key="port" value="443"/>
    <parameter key="days" value="30"/>
    <parameter key="server-name" value="${nodelabel}.example.com"/>
</service>
<monitor service="SSL-Cert-IMAPS-993" class-name=</pre>
"org.opennms.netmgt.poller.monitors.SSLCertMonitor" />
<monitor service="SSL-Cert-SMTPS-465" class-name=</pre>
"org.opennms.netmgt.poller.monitors.SSLCertMonitor" />
<monitor service="SSL-Cert-HTTPS-443" class-name=</pre>
"org.opennms.netmgt.poller.monitors.SSLCertMonitor" />
```

4.6.42. StrafePingMonitor

This monitor is used to monitor packet delay variation to a specific endpoint using *ICMP*. The main use case is to monitor a *WAN* end point and visualize packet loss and *ICMP* packet round trip time deviation. The *StrafePingMonitor* performs multiple *ICMP echo requests* (ping) and stores the response-time of each as well as the packet loss, in a *RRD* file. Credit is due to Tobias Oetiker, as this graphing feature is an adaptation of the SmokePing tool that he developed.

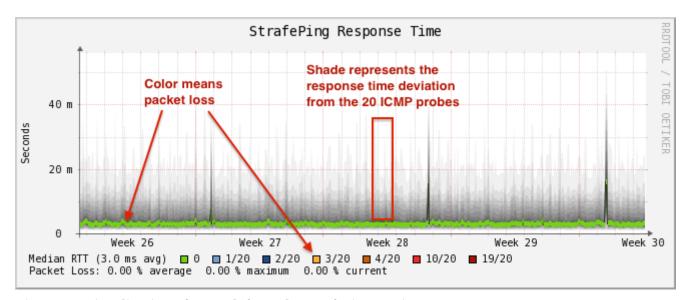


Figure 17. Visualization of a graph from the StrafePingMonitor

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.StrafePingMonitor |
|----------------|--|
| Remote Enabled | false |

Configuration and Usage

Monitor specific parameters for the StrafePingMonitor

| Parameter | Description | Requi red | Default value |
|-----------------------------|--|--------------|---------------|
| timeout | Time in milliseconds to wait before assuming that a packet has not responded | option al | 800 |
| retry | The number of retries to attempt when a packet fails to respond in the given timeout | option al | 2 |
| ping-count | The number of pings to attempt each interval | requir ed | 20 |
| failure- ping-count | The number of pings that need to fail for the service to be considered down | requir ed | 20 |
| allow- fragmentatio n | Whether to set the "Don't Fragment" bit on outgoing packets | option al | true |
| dscp | DSCP traffic-control value. | option al | 0 |
| packet-size | Number of bytes of the ICMP packet to send. | option al | 64 |
| wait- interval | Time in milliseconds to wait between each <i>ICMP echo-</i> request packet | requir ed | 50 |

| Parameter | Description | Requi red | Default value |
|--------------------|--|--------------|---|
| rrd- repository | The location to write <i>RRD data</i> . Generally, you will not want to change this from default | requir ed | <pre>\$0PENNMS_HOME/share/ rrd/response</pre> |
| rrd-base- name | The name of the RRD file to write (minus the extension, .rrd or .jrb) | requir ed | strafeping |

Examples

The *StrafePingMonitor* is typically used on WAN connections and not activated for every ICMP enabled device in your network. Further this monitor is much I/O heavier than just a simple RRD graph with a single ICMP response time measurement. By default you can find a separate *poller package* in the 'poller-configuration.xml' called *strafer*. Configure the include-range or a filter to enable monitoring for devices with the service *StrafePing*.



Don't forget to assign the service *StrafePing* on the IP interface to be activated.

The following example enables the monitoring for the service *StrafePing* on IP interfaces in the range 10.0.0.1 until 10.0.0.20. Additionally the Nodes have to be in a *surveillance category* named Latency.

```
<package name="strafer" >
  <filter>categoryName == 'Latency'</filter>
  <include-range begin="10.0.0.1" end="10.0.0.20"/>
  <rrd step="300">
    <rra>RRA:AVERAGE:0.5:1:2016
    <rra>RRA:AVFRAGF:0.5:12:1488
    <rra>RRA:AVERAGE:0.5:288:366</rra>
    <rra>RRA:MAX:0.5:288:366</rra>
    <rra>RRA:MIN:0.5:288:366</rra>
  <service name="StrafePing" interval="300000" user-defined="false" status="on">
     <parameter key="retry" value="0"/>
    <parameter key="timeout" value="3000"/>
    <parameter key="ping-count" value="20"/>
    <parameter key="failure-ping-count" value="20"/>
    <parameter key="wait-interval" value="50"/>
    <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response"/>
    <parameter key="rrd-base-name" value="strafeping"/>
  </service>
  <downtime interval="30000" begin="0" end="300000"/>
  <downtime interval="300000" begin="300000" end="43200000"/>
  <downtime interval="600000" begin="43200000" end="432000000"/>
  <downtime begin="432000000" delete="true"/>
</package>
<monitor service="StrafePing" class-name=</pre>
"org.opennms.netmgt.poller.monitors.StrafePingMonitor"/>
```

4.6.43. TcpMonitor

This monitor is used to test IP Layer 4 connectivity using TCP. The monitor establishes an TCP connection to a specific port. To test the availability of the service, the greetings banner of the application is evaluated. The behavior is similar to a simple test using the telnet command as shown in the example.

Simulating behavior of the monitor with telnet

```
root@vagrant:~# telnet 127.0.0.1 22
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2 ①
```

① Service greeting banner

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.TcpMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 65. Monitor specific parameters for the TcpMonitor

| Param eter | Description | Requi red | Default value |
|---------------|--|--------------|------------------|
| port | TCP port of the application. | requir ed | -1 |
| retry | Number of retries before the service is marked as <i>down</i> . | option al | 0 |
| timeout | Time in milliseconds to wait for the TCP service. | option al | 3000 |
| banner | Evaluation of the service connection banner with regular expression. By default any banner result is valid. | option al | * |

Examples

This example shows to test if the ICA service is available on TCP port 1494. The test evaluates the connection banner starting with ICA.

```
<service name="TCP-Citrix-ICA" interval="300000" user-defined="false" status="on">
    <parameter key="retry" value="0" />
    <parameter key="banner" value="ICA" />
    <parameter key="port" value="1494" />
    <parameter key="timeout" value="3000" />
    <parameter key="rrd-repository" value="/var/lib/opennms/rrd/response" />
    <parameter key="rrd-base-name" value="tcpCitrixIca" />
    <parameter key="ds-name" value="tcpCitrixIca" />
    </service>

<monitor service="TCP-Citrix-ICA" class-name=
    "org.opennms.netmgt.poller.monitors.TcpMonitor" />
```

4.6.44. SystemExecuteMonitor

If it is required to execute a system call or run a script to determine a service status, the SystemExecuteMonitor can be used. It is calling a script or system command, if required it provides additional arguments to the call. To determine the status of the service the SystemExecuteMonitor can rely on 0 or a non-0 exit code of system call. As an alternative, the output of the system call can be matched against a banner. If the banner is part of the output the status is interpreted as up. If the banner is not available in the output the status is determined as down.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.SystemExecuteMonitor |
|----------------|---|
| Remote Enabled | true |

Configuration and Usage

Table 66. Monitor specific parameters for the SystemExecuteMonitor

| Param eter | Description | Requi red | Default value |
|---------------|---|--------------|------------------|
| script | The system-call to execute. | requir ed | - |
| args | The arguments to hand over to the system-call. It supports variable replacement, see below. | option al | - |
| banner | A string that is match against the output of the system-call. If the output contains the banner, the service is determined as <i>UP</i> . | option al | - |

The parameter args supports variable replacement for the following set of variables.

Table 67. Variables which can be used in the configuration

| Variable | Description |
|-------------|--|
| \${timeout} | Timeout in milliseconds, based on config of the service. |

| Variable | Description |
|----------------|--|
| \${timeoutsec} | Timeout in seconds, based on config of the service. |
| \${retry} | Amount of retries based on config of the service. |
| \${svcname} | Service name based on the config of the service. |
| \${ipaddr} | IP-address of the interface the service is bound to. |
| \${nodeid} | Nodeid of the node the monitor is associated to. |
| \${nodelabel} | Nodelabel of the node the monitor is associated to. |

Examples

```
<parameter key="args" value="-i ${ipaddr} -t ${timeout}"/>
<parameter key="args" value="http://${nodelabel}/${svcname}/static"/>
```

SystemExecuteMonitor vs GpMonitor

The SystemExecuteMonitor is the successor of the GpMonitor. The main differences are:

- Variable replacement for the parameter args
- There are no fixed arguments handed to the system-call
- The SystemExecuteMonitor supports RemotePoller deployment

To migrate services from the *GpMonitor* to the *SystemExecuteMonitor* it is required to alter the parameter args. To match the arguments called hoption for the hostAddress and toption for the timeoutInSeconds. The args string that matches the *GpMonitor* call looks like this:

```
<parameter key="args" value="--hostname ${ipaddr} --timeout ${timeoutsec}" />
```

To migrate the GpMonitor parameters hoption and toption just replace the --hostname and --timeout directly in the args key.

4.6.45. VmwareCimMonitor

This monitor is part of the VMware integration provided in *Provisiond*. The monitor is specialized to test the health status provided from all *Host System* (host) sensor data.



This monitor is only executed if the host is in power state *on*.



This monitor requires to import hosts with *Provisiond* and the *VMware* import. OpenNMS Meridian requires network access to *VMware vCenter* and the hosts. To get the sensor data the credentials from *vmware-config.xml* for the responsible *vCenter* is used. The following asset fields are filled from *Provisiond* and is provided by *VMware* import feature: *VMware Management Server, VMware Managed Entity Type* and the *foreignId* which contains an internal *VMware vCenter Identifier*.

The global health status is evaluated by testing all available host sensors and evaluating the state of each sensor. A sensor state could be represented as the following:

- Unknown(0)
- OK(5)
- Degraded/Warning(10)
- Minor failure(15)
- Major failure(20)
- Critical failure(25)
- *Non-recoverable error(30)*

The service is up if **all** sensors have the status OK(5). If any sensor gives another status then OK(5) the service is marked as down. The monitor error reason contains a list of all sensors which not returned status OK(5).



In case of using Distributed Power Management the *standBy* state forces a service *down*. The health status is gathrered with a direct connection to the host and in stand by this connection is unavailable and the service is *down*. To deal with stand by states, the configuration *ignoreStandBy* can be used. In case of a stand by state, the service is considered as *up*.

state can be changed see the ignoreStandBy configuration parameter.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.VmwareCimMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

Table 68. Monitor specific parameters for the VmwareCimMonitor

| Parameter | Description | Require d | Default value |
|-----------|---|--------------|------------------|
| retry | Number of retries before the service is marked as down. | optional | 0 |
| timeout | Time in milliseconds to wait collecting the <i>CIM</i> sensor data. | optional | 3000 |

| Parameter | Description | Require d | Default value |
|---------------|---|--------------|------------------|
| ignoreStandBy | Treat power state <i>standBy</i> as <i>up</i> . | optional | false |

Examples

Some example configuration how to configure the monitor in the poller-configuration.xml.

4.6.46. VmwareMonitor

This monitor is part of the VMware integration provided in *Provisiond* and test the power state of a virtual machine (VM) or a host system (host). If the power state of a VM or host is *poweredOn* the service is *up*. The state *off* the service on the VM or Host is marked as *down*. By default *standBy* is also considered as *down*. In case of using Distributed Power Management the *standBy* state can be changed see the ignoreStandBy configuration parameter.



The information for the status of a virtual machine is collected from the responsible *VMware vCenter* using the credentials from the *vmware-config.xml*. It is also required to get specific asset fields assigned to an imported virtual machine and host system. The following asset fields are required, which are populated by the *VMware* integration in *Provisiond: VMware Management Server, VMware Managed Entity Type* and the *foreignId* which contains an internal *VMware vCenter Identifier*.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.VmwareMonitor |
|----------------|--|
| Remote Enabled | false |

Configuration and Usage

Table 69. Monitor specific parameters for the VmwareMonitor

| Parameter | Description | Requir ed | Default value |
|-----------|---|--------------|------------------|
| retry | Number of retries before the service is marked as <i>down</i> . | optiona l | 0 |

| Parameter | Description | Requir ed | Default value |
|-------------------|---|--------------|------------------|
| timeout | Time in milliseconds to wait for the <i>vCenter</i> to get the power state information. | optiona l | 3000 |
| ignoreStand By | Treat power state <i>standBy</i> as <i>up</i> . | optiona l | false |

Examples

Some example configuration how to configure the monitor in the poller-configuration.xml.

4.6.47. Win32ServiceMonitor

The Win32ServiceMonitor enables OpenNMS Meridian to monitor the running state of any Windows service. The service status is monitored using the Microsoft Windows® provided SNMP agent providing the LAN Manager MIB-II. For this reason it is required the SNMP agent and OpenNMS Meridian is correctly configured to allow queries against part of the *MIB* tree. The status of the service is monitored by polling the

```
svSvcOperatingState = 1.3.6.1.4.1.77.1.2.3.1.3
```

of a given service by the display name.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.Win32ServiceMonitor |
|----------------|--|
| Remote Enabled | false |

Configuration and Usage

Table 70. Monitor specific parameters for the Win32ServiceMonitor

| Parame ter | Description | Requi red | Default value |
|---------------|---|--------------|--------------------------|
| retry | Number of attempts to get the service state from SNMP agent | requir ed | From snmp- config.xml |

| Parame ter | Description | Requi red | Default value |
|------------------|---|--------------|--------------------------|
| timeout | Time in milliseconds to wait for the SNMP result before next attempt. | requir ed | From snmp- config.xml |
| service- name | The name of the service, this should be the exact name of the Windows service to monitor as it appears in the Services <i>MSC snap-in</i> . Short names such as you might use with net start will not work here. | requir ed | Server |



Non-English Windows The service-name is sometime encoded in languages other than English. Like in French, the *Task Scheduler* service is *Planificateur de tâche*. Because of the "â" (non-English character), the OID value is encoded in hexa (0x50 6C 61 6E 69 66 69 63 61 74 65 75 72 20 64 65 20 74 C3 A2 63 68 65 73).

Troubleshooting

If you've created a *Win32ServiceMonitor* poller and are having difficulties with it not being monitored properly on your hosts, chances are there is a difference in the name of the service you've created, and the actual name in the registry.

For example, I need to monitor a process called *Example Service* on one of our production servers. I retrieve the *Display name* from looking at the service in service manager, and create an entry in the poller-configuration.xml files using the exact name in the *Display name* field.

However, what I don't see is the errant space at the end of the service display name that is revealed when doing the following:

```
snmpwalk -v 2c -c <communitystring> <hostname> .1.3.6.1.4.1.77.1.2.3.1.1
```

This provides the critical piece of information I am missing:

```
iso.3.6.1.4.1.77.1.2.3.1.1.31.83.116.97.102.102.119.97.114.101.32.83.84.65.70.70.86.73
.69.87.32.66.97.99.107.103.114.111.117.110.100.32 = STRING: "Example Service "
```



Note the extra space before the close quote.

The extra space at the end of the name was difficult to notice in the service manager GUI, but is easily visible in the snmpwalk output. The right way to fix this would be to correct the service *Display name* field on the server, however, the intent of this procedure is to recommend verifying the true name using snmpwalk as opposed to relying on the service manager GUI.

Examples

Monitoring the service running state of the *Task Scheduler* on an English local Microsoft Windows® Server requires at minimum the following entry in the poller-configuration.xml.

4.6.48. WsManMonitor

This monitor can be used to issue a WS-Man *Get* command and validate the results using a SPEL expression.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.WsManMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

Table 71. Monitor specific parameters for the WsManMonitor

| Parameter | Description | Require d | Default value |
|------------------|---|--------------|------------------|
| resource- uri | Resource URI | require d | - |
| rule | SPEL expression applied against the result of the <i>Get</i> | require d | - |
| selector. | Used to filter the result set. All selectors must prefixed with selector. | optional | (None) |

Examples

The following monitor will issue a *Get* against the configured resource and verify that the correct service tag is returned:

4.6.49. XmpMonitor

The XMP monitor tests for XMP service/agent availability by establishing an XMP session and querying the target agent's sysObjectID variable contained in the Core MIB. The service is considered available when the session attempt succeeds and the agent returns its sysObjectID without error.

Monitor facts

| Class Name | org.opennms.netmgt.poller.monitors.XmpMonitor |
|----------------|---|
| Remote Enabled | false |

Configuration and Usage

These parameters can be set in the *XMP* service entry in *collectd-configuration.xml* and will override settings from *xmp-config.xml*. Also, don't forget to add an entry in *response-graph.properties* so that response values will be graphed.

Table 72. Monitor specific parameters for the XmpMonitor

| Paramete r | Description | Require d | Default value |
|---------------|--|--------------|------------------|
| timeout | Time in milliseconds to wait for a successful session. | optional | 5000 |
| authenUser | The authenUser parameter for use with the XMP session. | optional | xmpUser |
| port | TCP port to connect to for XMP session establishment | optional | 5270 |
| mib | Name of MIB to query | optional | core |
| object | Name of MIB object to query | optional | sysObjectID |

Examples

Adding entry in collectd-configuration.xml

Add entry in response-graph.properties

```
reports=icmp, \
xmp, \ . . . .
report.xmp.name=XMP
report.xmp.columns=xmp
report.xmp.type=responseTime
report.xmp.command=--title="XMP Response Time" \
--vertical-label="Seconds" \
DEF:rtMills={rrd1}:xmp:AVERAGE \
DEF:minRtMills={rrd1}:xmp:MIN \
DEF:maxRtMills={rrd1}:xmp:MAX \
CDEF:rt=rtMills,1000,/\
CDEF:minRt=minRtMills,1000,/ \
CDEF:maxRt=maxRtMills,1000,/ \
LINE1:rt#0000ff:"Response Time" \
GPRINT:rt:AVERAGE:" Avg \\: %8.21f %s" \
GPRINT:rt:MIN:"Min \\: %8.2lf %s" \
GPRINT:rt:MAX:"Max \\: %8.21f %s\\n"
```

Chapter 5. Performance Management

In *OpenNMS* collection of performance data is done by the *Collectd* daemon. *Management Agents* and protocols to access performance data is implemented in *Collectors*. These *Collectors* are scheduled and run in parallel in a global defined *Thread Pool* in *Collectd*.

This section describes how to configure *Collectd* for performance data collection with all available *Collectors* coming with *OpenNMS Meridian*.

5.1. Collectd Configuration

Table 73. Configuration and log files related to Collectd

| File | Description |
|---|---|
| <pre>\$OPENNMS_HOME/etc/collectd- configuration.xml</pre> | Configuration file for global <i>Collectd</i> daemon and <i>Collectors</i> configuration |
| <pre>\$OPENNMS_HOME/logs/collectd.l og</pre> | Log file for all <i>Collectors</i> and the global <i>Collectd</i> daemon |
| <pre>\$OPENNMS_HOME/etc/snmp- graph.properties</pre> | RRD graph definitions to render performance data measurements in the Web UI |
| <pre>\$OPENNMS_HOME/etc/snmp- graph.properties.d</pre> | Directory with <i>RRD</i> graph definitions for devices and applications to render performance data measurements in the <i>Web UI</i> |
| <pre>\$OPENNMS_HOME/etc/events/open nms.events.xml</pre> | Event definitions for <i>Collectd</i> , i.e. <i>dataCollectionSucceeded</i> , and <i>dataCollectionFailed</i> |

To change the behavior for performance data collection, the collectd-configuration.xml file can be modified. The configuration file is structured in the following parts:

- Global daemon config: Define the size of the used Thread Pool to run Collectors in parallel.
- *Collection packages*: Packages to allow the grouping of configuration parameters for *Collectors*.
- *Collection service association*: Based on the name of the collection service, the implementation for application or network management protocols are assigned.

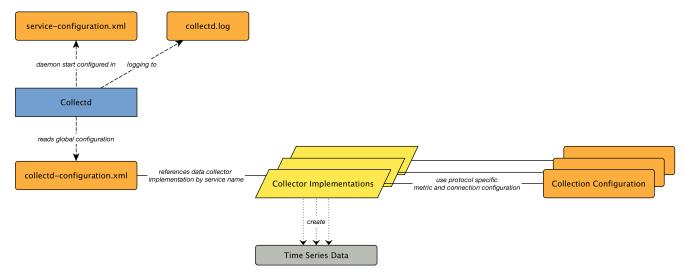


Figure 18. Collectd overview for associated files and configuration

The global behavior, especially the size of the *Thread Pool* for *Collectd*, is configured in the collectd-configuration.xml.

Global configuration parameters for Collectd

```
<collectd-configuration
    threads="50"> ①
```

① Size of the *Thread Pool* to run *Collectors* in parallel

5.2. Collection Packages

To define more complex collection configuration it is possible to group *Service* configurations which provide performance metrics into *Collection Packages*. They allow to assign to *Nodes* different *Service Configurations* to differentiate collection of performance metrics and connection settings. To assign a *Collection Package* to nodes the Rules/Filters syntax can be used.

Multiple packages can be configured, and an interface can exist in more than one package. This gives great flexibility how the service levels will be determined for a given device. The order how *Collection Packages* are defined is important when *IP Interfaces* match multiple *Collection Packages* with the same *Service* configuration. The last *Collection Package* on the service will be applied. This can be used to define a less specific catch all filter for a default configuration. A more specific *Collection Package* can be used to overwrite the default setting.

Collection Package Attributes

- 1 Unique name of the collection package.
- ② Apply this package to all *IP interfaces* with a configured *IPv4* address (not equal 0.0.0.0)

- 3 Evaluate IPv4 rule to collect for all IPv4 interfaces in the given range
- 4 Evaluate IPv6 rule to collect for all IPv6 interfaces in the given range

5.2.1. Service Configurations

Service Configurations define what Collector to use and which performance metrics needs to be collected. Service Configurations contains common Service Attributes as well as Collector specific parameters.

Service Configuration Attributes

- ① Service Configuration name which is mapped to a specific *Collector* implementation.
- ② The interval at which the service is to be collected. (in milliseconds).
- 3 Marker to say if service is user defined, used specifically for UI purposes.
- 4 Service is collected only if on.
- (5) Assign performance data collection metric groups named default.
- 6 Enable threshold evaluation for metrics provided by this service.
- 7 Run the SnmpCollector implementation for the service named SNMP

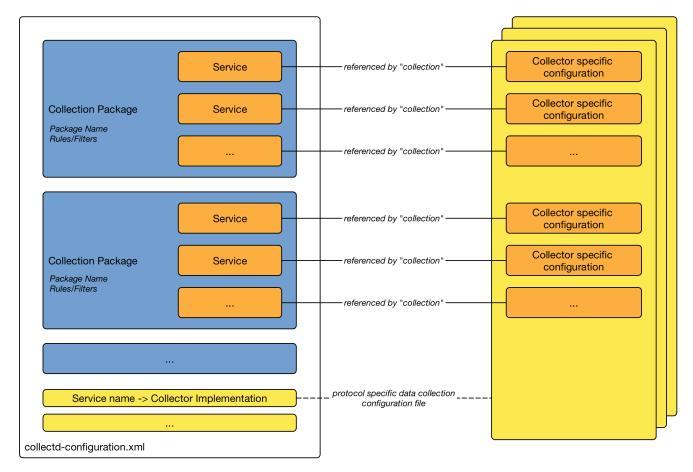


Figure 19. Configuration overview for data collection with Collectd

5.3. Collectors

5.3.1. JmxCollector

The *JmxCollector* is used to collect performance data via *JMX*. Attributes are extracted from the available *MBeans*.

Collector Facts

| Class Name | org.opennms.netmgt.collectd.Jsr160Collector |
|------------|---|
| Package | core |

Collector Parameters

Table 74. Collector specific parameters for the Jsr160Collector

| Parameter | Description | Requ ired | Default value |
|--------------------------|---|--------------|---------------|
| collection | The name of the <i>JMX Collection</i> to use | requi red | (none) |
| thresholdin g-enabled | Whether collected performance data shall be tested against thresholds | optio nal | true |

| Parameter | Description | Requ ired | Default value |
|-------------------|---|--------------|---|
| retry | Number of retries | optio nal | 3 |
| friendlyNam e | Name of the path in which the metrics should be stored | optio nal | Value of the port, or 'jsr160' if no port is set. |
| factory | The password strategy to use. Supported values are: STANDARD (for authentication), PASSWORD_CLEAR (same as STANDARD) and SASL (if secure connection is required) | optio nal | STANDARD |
| url | The connection url, e.g. service:jmx:rmi:localhost:18980. The ip address can be substituted. Use \${ipaddr} in that case, e.g.: service:jmx:rmi:\${ipaddr}:18980 | optio nal | (none) |
| username | The username if authentication is required. | optio nal | (none) |
| password | The password if authentication is required. | optio nal | (none) |
| port | Deprecated. JMX port. | optio nal | 1099 |
| protocol | Deprecated. Protocol used in the JMX connection string. | optio nal | rmi |
| urlPath | Deprecated . Path used in JMX connection string. | optio nal | /jmxrmi |
| rmiServerPo rt | Deprecated. RMI port. | optio nal | 45444 |
| remoteJMX | Deprecated . Use an alternative JMX URL scheme. | optio nal | false |



The parameters port, protocol, urlPath, rmiServerPort and remoteJMX are deprecated and should be replaced with the url parameter. If url is not defined the collector falls back to *Legacy Mode* and the deprecated parameters are used instead to build the connection url.



If a service requires different configuration it can be overwritten with an entry in \$OPENNMS_HOME/etc/jmx-config.xml.

JMX Collection Configuration

JMX Collections are defined in the etc/jmx-datacollection-config.xml and etc/jmx-datacollection-config.d/.

Here is a snippet providing a collection definition named opennms-poller:

3rd Party JMX Services

Some java applications provide their own JMX implementation and require certain libraries to be present on the classpath, e.g. the java application server Wildfly. In order to successfully collect data the following steps may be required:

- Place the jmx client lib to the \$OPENNMS_HOME/lib folder (e.g. jboss-cli-client.jar)
- Configure the JMX-Collector accordingly (see below)
- Configure the collection accordingly (see above)

Example

5.3.2. SnmpCollector

The *SnmpCollector* is used to collect performance data through the *SNMP protocol*. Access to the *SNMP Agent* is configured through the *SNMP configuration* in the *Web User Interface*.

Collector Facts

```
Class Name org.opennms.netmgt.collectd.SnmpCollector
```

Collector Parameters

Table 75. Collector specific parameters for the SnmpCollector

| Parameter | Description | Requir ed | Default value |
|--------------------------|--|--------------|--------------------|
| collection | The name of the SNMP Collection to use. | require d | default |
| thresholding- enabled | Whether collected performance data shall be tested against thresholds. | optiona l | true |
| timeout | Timeout in milliseconds to wait for SNMP responses. | optiona l | SNMP configuration |

SNMP Collection Configuration

SNMP Collection are defined in the etc/datacollection-config.xml and etc/datacollection.d/*.xml files.

```
<?xml version="1.0"?>
<datacollection-config rrd-repository="/var/lib/opennms/rrd/snmp/"> ①
   <snmp-collection name="default" ②</pre>
                    snmpStorageFlag="select"> 3
       <rrd step="300"> 4
           <rra>RRA:AVERAGE:0.5:1:2016
           <rra>RRA:AVERAGE:0.5:12:1488
           <rra>RRA:AVERAGE:0.5:288:366
           <rra>RRA:MAX:0.5:288:366</rra>
           <rra>RRA:MIN:0.5:288:366</rra>
       </rrd>
       <include-collection dataCollectionGroup="MIB2"/> ⑤
       <include-collection dataCollectionGroup="3Com"/>
       <include-collection dataCollectionGroup="VMware-Cim"/>
   </snmp-collection>
</datacollection-config>
```

- ① Directory where to persist *RRD* files on the file system, ignored if *NewTS* is used as time series storage.
- ② Name of the *SNMP* data collection referenced in the *Collection Package* in collectd-configuration.xml.
- ③ Configure SNMP MIB-II interface metric collection behavior: *all* means collect metrics from all interfaces, *primary* only from interface provisioned as *primary* interface, *select* only from

manualy selected interfaces from the Web UI.

- ④ RRD archive configuration for this set of performance metrics, ignored when NewTS is used as time series storage.
- ⑤ Include device or application specific performance metric OIDS to collect.

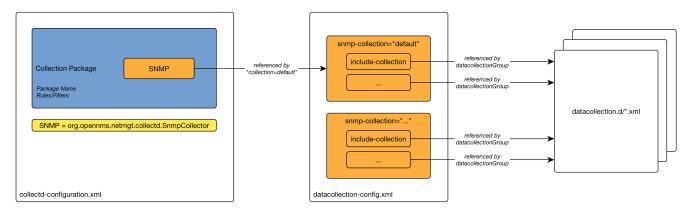


Figure 20. Configuration overview for SNMP data collection

5.3.3. WS-Management

Web Services-Management (WS-Management) is a DMTF open standard defining a SOAP-based protocol for the management of servers, devices, applications and various Web services. Windows Remote Management (WinRM) is the Microsoft implementation of WS-Management Protocol. OpenNMS Meridian currently provides support for detecting, polling and collecting metrics from WS-Man endpoints.

Setup

Before setting up OpenNMS Meridian to communicate with a WS-Management agent, you should confirm that it is properly configured and reachable from the OpenNMS Meridian system. If you need help enabling the WS-Management agent, consult the documentation from the manufacturer. Here are some link resources that could help:

- Installation and Configuration for Windows Remote Management
- Troubleshooting WinRM connection and authentication

We suggest using the Openwsman command line client for validating authentication and connectivity. Packages are available for most distributions under wsmancli.

For example:

```
wsman identify -h localhost -P 5985 -u wsman -p secret
```

Once validated, add the agent specific details to the OpenNMS Meridian configuration, defined in the next section.

Troubleshooting and Commands

For troubleshooting there is a set of commands you can use in Powershell verified on Microsoft

Windows Server 2012.

Enable WinRM in PowerShell

Enable-PSRemoting

Setup Firewall for WinRM over HTTP

netsh advfirewall firewall add rule name="WinRM-HTTP" dir=in localport=5985
protocol=TCP action=allow

Setup Firewall for WinRM over HTTPS

netsh advfirewall firewall add rule name="WinRM-HTTPS" dir=in localport=5986
protocol=TCP action=allow

Test WinRM on local Windows Server

winrm id

Show WinRM configuration on Windows Server

winrm get winrm/config

Show listener for configuration on Windows Server

winrm e winrm/config/listener

Test connectivity from a Linux system

```
nc -z -w1 <windows-server-ip-or-host> 5985;echo $?
```



Use BasicAuthentication just with *WinRM* over *HTTPS* with verifiable certificates in production environment.

Enable BasicAuthentication

```
winrm set winrm/config/client/auth '@{Basic="true"}'
winrm set winrm/config/service/auth '@{Basic="true"}'
winrm set winrm/config/service '@{AllowUnencrypted="true"}'
```

Agent Configuration

The agent specific configuration details are maintained in etc/wsman-config.xml. This file has a similar structure as etc/snmp-config.xml, which the reader may already be familiar with.

This file is consulted when a connection to a WS-Man Agent is made. If the IP address of the agent is matched by the range, specific or ip-match elements of a definition, then the attributes on that definition are used to connect to the agent. Otherwise, the attributes on the outer wsman-config definition are used.

This etc/wsman-config.xml files is automatically reloaded when modified.

Here is an example with several definitions:

Table 76. Collector configuration attributes

| Attribute | Description | Default |
|---------------------|--|----------------------|
| timeout | HTTP Connection and response timeout in milliseconds. | HTTP client default |
| retry | Number of retries on connection failure. | 0 |
| username | Username for basic authentication. | none |
| password | Password used for basic authentication. | none |
| port | HTTP/S port | Default for protocol |
| max- elements | Maximum number of elements to retrieve in a single request. | no limit |
| ssl | Enable SSL | False |
| strict-ssl | Enforce SSL certificate verification. | True |
| path | Path in the URL to the WS-Management service. | / |
| product- vendor | Used to overwrite the detected product vendor. | none |
| product- version | Used to overwrite the detected product version. | none |
| gss-auth | Enables GSS authentication. When enabled a reverse lookup is performed on the target IP address in order to determine the canonical host name. | False |



If you try to connect against *Microsoft Windows Server* make sure to set specific ports for *WinRM* connections. By default *Microsoft Windows Server* uses port TCP/5985 for plain text and port TCP/5986 for *SSL* connections.

Collector

Configuration for the WS-Management collector is stored in etc/wsman-datacollection-config.xml and etc/wsman-datacollection.d/*.xml.



The contents of these files are automatically merged and reloaded when changed. The default WS-Management collection looks as follows:

The magic happens with the <include-all-system-definitions/> element which automatically includes all of the system definitions into the collection group.



If required, you can include a specific system-definition with <include-system-definition>sys-def-name</include-system-definition>.

System definitions and related groups can be defined in the root etc/wsman-datacollection-config.xml file, but it is preferred that be added to a device specific configuration files in etc/wsman-datacollection-config.d/*.xml.



Avoid modifying any of the distribution configuration files and create new ones to store you specific details instead.

Here is an example configuration file for a *Dell iDRAC*:

```
<?xml version="1.0"?>
<wsman-datacollection-config>
    <group name="drac-system"</pre>
           resource-uri="http://schemas.dell.com/wbem/wscim/1/cim-
schema/2/root/dcim/DCIM_ComputerSystem"
           resource-type="node">
        <attrib name="OtherIdentifyingInfo" index-of="#IdentifyingDescriptions matches"
'.*ServiceTag'" alias="serviceTag" type="String"/>
    </group>
    <group name="drac-power-supply"</pre>
           resource-uri="http://schemas.dmtf.org/wbem/wscim/1/*"
           dialect="http://schemas.microsoft.com/wbem/wsman/1/WQL"
           filter="select
InputVoltage, InstanceID, PrimaryStatus, SerialNumber, TotalOutputPower from
DCIM_PowerSupplyView where DetailedState != 'Absent'"
           resource-type="dracPowerSupplyIndex">
        <attrib name="InputVoltage" alias="inputVoltage" type="Gauge"/>
        <attrib name="InstanceID" alias="instanceId" type="String"/>
        <attrib name="PrimaryStatus" alias="primaryStatus" type="Gauge"/>
        <attrib name="SerialNumber" alias="serialNumber" type="String"/>
        <attrib name="TotalOutputPower" alias="totalOutputPower" type="Gauge"/>
    </group>
    <system-definition name="Dell iDRAC (All Version)">
        <rule>#productVendor matches '^Dell.*' and #productVersion matches
'.*iDRAC.*'</rule>
        <include-group>drac-system</include-group>
        <include-group>drac-power-supply</include-group>
    </system-definition>
</wsman-datacollection-config>
```

System Definitions

Rules in the system definition are written using SpEL expressions.

The expression has access to the following variables in it `s evaluation context:

| Name | Туре |
|----------------|---|
| (root) | org.opennms.netmgt.model.OnmsNode |
| agent | org.opennms.netmgt.collection.api.CollectionAgent |
| productVendor | java.lang.String |
| productVersion | java.lang.String |

If a particular agent is matched by any of the rules, then the collector will attempt to collect the referenced groups from the agent.

Group Definitions

Groups are retrieved by issuing an Enumerate command against a particular Resource URI and parsing the results. The Enumerate commands can include an optional filter in order to filter the records and attributes that are returned.



When configuring a filter, you must also specify the dialect.

The resource type used by the group must of be of type node or a generic resource type. Interface level resources are not supported.

When using a generic resource type, the IndexStorageStrategy cannot be used since records have no implicit index. Instead, you must use an alternative such as the SiblingColumnStorageStrategy.

If a record includes a multi-valued key, you can collect the value at a specific index with an index-of expression. This is best demonstrated with an example. Let's assume we wanted to collect the ServiceTag from the following record:

```
<IdentifyingDescriptions>CIM:GUID</IdentifyingDescriptions>
<IdentifyingDescriptions>CIM:Tag</IdentifyingDescriptions>
<IdentifyingDescriptions>DCIM:ServiceTag</IdentifyingDescriptions>
<OtherIdentifyingInfo>45454C4C-3700-104A-8052-C3C01BB25031</OtherIdentifyingInfo>
<OtherIdentifyingInfo>mainsystemchassis</OtherIdentifyingInfo>
<OtherIdentifyingInfo>C8BBBP1</OtherIdentifyingInfo>
```

Specifying, the attribute name OtherIdentifyingInfo would not be sufficient, since there are multiple values for that key. Instead, we want to retrieve the value for the OtherIdentifyingInfo key at the same index where IdentifyingDescriptions is set to DCIM:ServiceTag.

This can be achieved using the following attribute definition:

```
<attrib name="OtherIdentifyingInfo" index-of="#IdentifyingDescriptions matches
'.*ServiceTag'" alias="serviceTag" type="String"/>
```

Detector

The WS-Management detector attempts to connect to the agent defined in wsman-config.xml and issues an Identify command. If a valid response is received, the product vendor and product version are stored in the vendor and modelNumber fields of the associated node`s assets table.

For example, a Windows Server 2008 machine returns:

| Product Vendor | Microsoft Corporation |
|-----------------|---------------------------------|
| Product Version | OS: 6.1.7601 SP: 1.0 Stack: 2.0 |

If these assets field are being used for another purpose, this behavior can be disabled by settings the updateAssets parameters to false in the detector configuration of the appropriate foreign

source.



Some agents may respond to the Identify command with generic identities such as Openwsman 2.0.0. These values can be overridden by specifying the product-vendor and product-version attributes in wsman-config.xml.

Example detector configuration:

The response is logged as *DEBUG* information in provisiond.log and looks like the following:

```
ID: 3
Response-Code: 200
309Encoding: UTF-8
Content-Type: application/soap+xml; charset=UTF-8
Headers: {Content-Length=[787], content-type=[application/soap+xml;charset=UTF-8],
Date=[Mon, 08 Feb 2016 14:21:20 GMT], Server=[Microsoft-HTTPAPI/2.0]}
Payload:
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xml:lang="en-US">
 <s:Header/>
 <s:Body>
    <wsmid:IdentifyResponse xmlns:wsmid=</pre>
"http://schemas.dmtf.org/wbem/wsman/identity/1/wsmanidentity.xsd">
    <wsmid:ProtocolVersion>
http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd</wsmid:ProtocolVersion>
    <wsmid:ProductVendor>Microsoft Corporation</wsmid:ProductVendor> ①
    <wsmid:ProductVersion>OS: 6.2.9200 SP: 0.0 Stack: 3.0/wsmid:ProductVersion> ②
    <wsmid:SecurityProfiles>
<wsmid:SecurityProfileName>http://schemas.dmtf.org/wbem/wsman/1/wsman/secprofile/http/
basic</wsmid:SecurityProfileName>
<wsmid:SecurityProfileName>http://schemas.dmtf.org/wbem/wsman/1/wsman/secprofile/http/
spnego-kerberos</wsmid:SecurityProfileName>
    </wsmid:SecurityProfiles>
    </wsmid:IdentifyResponse>
 </s:Body>
</s:Envelope>
```

- 1 ProductVendor: Stored to the asset field vendor
- ② ProductVersion: Stored in the asset field modelNumber

1

The information of the asset fields are used in the *System Definition Rule* to decide which performance metrics will be gathered from *Collectd*.

5.4. Stress Testing

5.5. Stress Testing

The metrics:stress *Karaf Shell* command can be used to simulate load on the active persistence strategy, whether it be RRDtool, JRobin, or Newts.

The tool works by generating collection sets, similar to those built when performing data collection, and sending these to the active persistence layer. By using the active persistence layer, we ensure that we use the same write path which is used by the actual data collection services.

To get started, log into the *Karaf Shell* on your system:

```
ssh -p 8101 admin@localhost
```

Generate samples for 10 nodes every 15 seconds and printing the statistic report every 30 seconds:

```
metrics:stress -n 10 -i 15 -r 30
```

While active, the command will continue to generate and persist collection sets. During this time you can monitor the system I/O and other relevant statistics.

When your done, use **CTRL+C** to stop the stress tool.

A complete list of options is available using:

```
metrics:stress --help
```

5.5.1. Interpreting the output

The statistics output by the tool can be be interpreted as follows:

numeric-attributes-generated

The number of numeric attributes that were sent to the persistence layer. We have no guarantee as to whether or not these were actually persisted.

string-attributes-generated

The number of string attributes that were sent to the persistence layer. We have no guarantee as to whether or not these were actually persisted.

batches

The count is used to indicate how many batches of collection sets (one at every interval) were

| sent to the persistence layer. The timers show how much time was spent generating the batch and sending the batch to the persistence layer. |
|---|
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

Chapter 6. Events

Events are central to the operation of the OpenNMS Meridian platform, so it's critical to have a firm grasp of this topic.



Whenever something in OpenNMS Meridian appears to work by magic, it's probably events working behind the curtain.

6.1. Anatomy of an Event

Events are structured historical records of things that happen in OpenNMS Meridian and the nodes, interfaces, and services it manages. Every event has a number of fixed **fields** and zero or more **parameters**.

Mandatory Fields

UEI (Universal Event Identifier)

A string uniquely identifying the event's type. UEIs are typically formatted in the style of a URI, but the only requirement is that they start with the string uei..

Event Label

A short, static label summarizing the gist of all instances of this event.

Description

A long-form description describing all instances of this event.

Log Message

A long-form log message describing this event, optionally including expansions of fields and parameters so that the value is tailored to the event at hand.

Severity

A severity for this event type. Possible values range from Cleared to Critical.

Event ID

A numeric identifier used to look up a specific event in the OpenNMS Meridian system.

Notable Optional Fields

Operator Instruction

A set of instructions for an operator to respond appropriately to an event of this type.

Alarm Data

If this field is provided for an event, OpenNMS Meridian will create, update, or clear **alarms** for events of that type according to the alarm-data specifics.

6.2. Sources of Events

Events may originate within OpenNMS Meridian itself or from outside.

Internally-generated events can be the result of the platform's monitoring and management functions (*e.g.* a monitored node becoming totally unavailable results in an event with the UEI uei.opennms.org/nodes/nodeDown) or they may act as inputs or outputs of housekeeping processes.

The following subsections summarize the mechanisms by which externally-created events can arrive.

6.2.1. SNMP Traps

If SNMP-capable devices in the network are configured to send **traps** to OpenNMS Meridian, these traps are transformed into events according to pre-configured rules. The Trapd service daemon, which enables OpenNMS Meridian to receive SNMP traps, is enabled by default.



Disabling the Trapd service daemon will render OpenNMS Meridian **incapable** of receiving SNMP traps.

Event definitions are included with OpenNMS Meridian for traps from many vendors' equipment.

6.2.2. Syslog Messages

Syslog messages sent over the network to OpenNMS Meridian can be transformed into events according to pre-configured rules.



The Syslogd service daemon, which enables OpenNMS Meridian to receive syslog messages over the network, must be enabled for this functionality to work. This service daemon is **disabled** by default.

Parsers

Different parsers can be used to convert the syslog message fields into OpenNMS Meridian event fields.

| Parser | Description |
|--|--|
| org.opennms.netmgt.syslogd.CustomSyslogParser | Default parser that uses a regex statement to parse the syslog header. |
| org.opennms.netmgt.syslogd.RadixTre eSyslogParser | Parser that uses an internal list of <i>grok</i> -style statements to parse the syslog header. |
| org.opennms.netmgt.syslogd.SyslogNG Parser | Parser that strictly parses messages in the default pattern of syslog-ng. |
| org.opennms.netmgt.syslogd.Rfc5424S yslogParser | Parser that strictly parses the RFC 5424 format for syslog messages. |

RadixTreeSyslogParser

The RadixTreeSyslogParser normally uses a set of internally-defined patterns to parse multiple syslog message formats. If you wish to customize the set of patterns, you can put a new set of patterns into a syslog-grok-patterns.txt in the etc directory for OpenNMS Meridian.

The patterns are defined in *grok*-style statements where each token is defined by a **%{PATTERN:semantic}** clause. Whitespace in the pattern will match 0...n whitespace characters and character literals in the pattern will match the corresponding characters. The '%' character literal must be escaped by using a backslash, ie. '\%'.



The RadixTreeSyslogParser's *grok* implementation only supports a limited number of pattern types. However, these patterns should be sufficient to parse any syslog message format.

The patterns should be arranged in the file from most specific to least specific since the first pattern to successfully match the syslog message will be used to construct the OpenNMS Meridian event.

| Patter n | Description |
|-------------|---|
| INT | Positive integer. |
| MONTH | 3-character English month abbreviation. |
| NOSPAC E | String that contains no whitespace. |
| STRING | String. Because this matches any character, it must be followed by a delimiter in the pattern string. |

| Semantic Token | Description |
|------------------|---|
| day | 2-digit day of month (1-31). |
| facilityPriority | Facility-priority integer. |
| hostname | String hostname (unqualified or FQDN), IPv4 address, or IPv6 address. |
| hour | 2-digit hour of day (0-23). |
| message | Remaining string message. |
| messageId | String message ID. |
| minute | 2-digit minute (0-59). |
| month | 2-digit month (1-12). |
| processId | String process ID. |
| processName | String process name. |
| second | 2-digit second (0-59). |
| secondFraction | 1- to 6-digit fractional second value as a string. |
| timezone | String timezone value. |
| version | Version. |
| year | 4-digit year. |

6.2.3. TL1 Autonomous Messages

Autonomous messages can be retrieved from certain TL1-enabled equipment and transformed into events.



The Tl1d service daemon, which enables OpenNMS Meridian to receive TL1 autonomous messages, must be enabled for this functionality to work. This service daemon is **disabled** by default.

6.2.4. XML-TCP

Any application or script can create custom events in OpenNMS Meridian by sending properly-formatted XML data over a TCP socket.

6.2.5. ReST

Posting an event in XML format to the appropriate endpoint in the OpenNMS Meridian ReST API will cause the creation of a corresponding event, just as with the XML-TCP interface.

6.3. The Event Bus

At the heart of OpenNMS Meridian lies an **event bus**. Any OpenNMS Meridian component can *publish* events to the bus, and any component can *subscribe* to receive events of interest that have been published on the bus. This publish-subscribe model enables components to use events as a mechanism to send messages to each other. For example, the provisioning subsystem of OpenNMS Meridian publishes a *node-added* event whenever a new node is added to the system. Other subsystems with an interest in new nodes subscribe to the *node-added* event and automatically receive these events, so they know to start monitoring and managing the new node if their configuration dictates. The publisher and subscriber components do not need to have any knowledge of each other, allowing for a clean division of labor and lessening the programming burden to add entirely new OpenNMS Meridian subsystems or modify the behavior of existing ones.

6.3.1. Associate an Event to a given node

There are 2 ways to associate an existing node to a given event prior sending it to the Event Bus:

- Set the **nodeId** of the node in question to the event.
- For requisitioned nodes, set the _foreignSource and _foreignId as parameters to the event. Then, any incoming event without a nodeId and these 2 parameters will trigger a lookup on the DB; if a node is found, the nodeId attribute will be dynamically set into the event, regardless which method has been used to send it to the Event Bus.

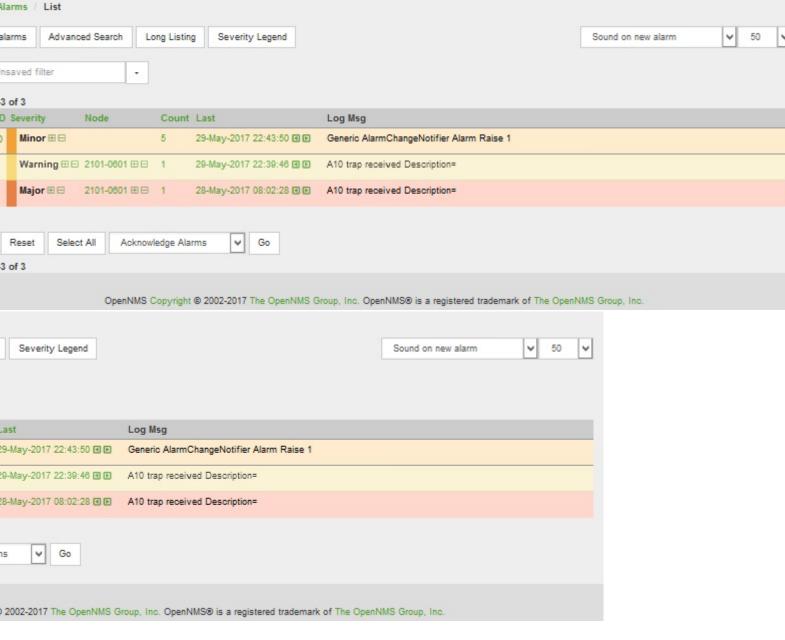
Chapter 7. Alarms

7.1. Alarm Sounds

Often users want an audible indication of a change in alarm state. The *OpenNMS Meridian* alarm list page has the optional ability to generate a sound either on each new alarm or (more annoyingly) on each change to an alarm event count on the page.

The figure Alarm Sounds View shows the alarm list page when alarms sounds are enabled.

Alarm Sounds View



By default the alarm sound feature is disabled. System Administrators must activate the sound feature and also set the default sound setting for all users. However users can modify the default sound setting for the duration of their logged-in session using a drop down menu with the following options:

- Sound off: no sounds generated by the page.
- Sound on new alarm: sounds generated for every new alarm on the page.
- Sound on new alarm count: sounds generated for every increase in alarm event count for alarms on the page.

7.2. Flashing Unacknowledged Alarms

By default *OpenNMS Meridian* displays the alarm list page with acknowledged and unacknowledged alarms listed in separate search tabs. In a number of operational environments it is useful to see all of the alarms on the same page with unacknowledged alarms flashing to indicate that they haven't yet been noticed by one of the team. This allows everyone to see at a glance the real time status of all alarms and which alarms still need attention.

The figure Alarm Sounds View also shows the alarm list page when flashing unacknowledged alarms are enabled. Alarms which are unacknowledged flash steadily. Alarms which have been acknowledged do not flash and also have a small tick beside the selection check box. All alarms can be selected to be escalated, cleared, acknowledged and unacknowledged.

7.3. Configuring Alarm Sounds and Flashing

By default *OpenNMS Meridian* does not enable alarm sounds or flashing alarms. The default settings are included in opennms.properties. However rather than editing the default opennms.properties file, the system administrator should enable these features by creating a new file in opennms.properties.d and applying the following settings;

\${OPENNMS_HOME}/etc/opennms.properties.d/alarm.listpage.properties

```
# ##### Alarm List Page Options ######
# Several options are available to change the default behaviour of the Alarm List
Page.
# <opennms url>/opennms/alarm/list.htm
# The alarm list page has the ability to generate a sound either on each new alarm
# or (more annoyingly) on each change to an alarm event count on the page.
# Turn on the sound feature. Set true and Alarm List Pages can generate sounds in the
web browser.
opennms.alarmlist.sound.enable=true
# Set the default setting for how the Alarm List Pages generates sounds. The default
setting can be
# modified by users for the duration of their logged-in session using a drop down menu
#
    off = no sounds generated by the page.
#
     newalarm = sounds generated for every new alarm in the page
     newalarmcount = sounds generated for every increase in alarm event count for
alarms on the page
opennms.alarmlist.sound.status=off
# By default the alarm list page displays acknowledged and unacknowledged alarms in
separate search tabs
# Some users have asked to be able to see both on the same page. This option allows
the alarm list page
# to display acknowledged and unacknowledged alarms on the same list but
unacknowledged alarms
# flash until they are acknowledged.
opennms.alarmlist.unackflash=true
```

The sound played is determined by the contents of the following file \${OPENNMS_HOME}/jetty-webapps/opennms/sounds/alert.wav

If you want to change the sound, create a new wav file with your desired sound, name it alert.wav and replace the default file in the same directory.

Chapter 8. Notifications

8.1. Introduction

OpenNMS Meridian uses notifications to make users aware of an event. Common notification methods are email and paging, but notification mechanisms also exist for:

- · Arbitrary HTTP GET and POST operations
- Arbitrary external commands
- Asterisk call origination
- IRCcat Internet Relay Chat bot
- SNMP Traps
- Slack, Mattermost, and other API-compatible team chat platforms
- Twitter, GNU Social, and other API-compatible microblog services
- User-provided scripts in any JSR-223 compatible language
- XMPP

The notification daemon *Notifd* creates and sends notifications according to configured rules when selected events occur in *OpenNMS Meridian*.

8.2. Getting Started

The status of notifications is indicated by an icon at the top right of the web UI's navigation bar. *OpenNMS Meridian* installs with notifications globally disabled by default.

8.2.1. Enabling Notifications

To enable notifications in *OpenNMS Meridian*, log in to the web UI as a user with administrator privileges. Hover over the user icon and click the *Configure OpenNMS* link. The controls for global notification status appear in the top-level configuration menu as *Notification Status*. Click the *On* radio button and then the *Update* button. Notifications are now globally enabled.



The web workflow above is functionally equivalent to editing the notifd-configuration.xml file and setting status="on" in the top-level notifd-configuration element. This configuration file change is picked up on the fly with no need to restart or send an event.

8.2.2. Configuring Destination Paths

To configure notification destination paths in *OpenNMS Meridian*, navigate to *Configure OpenNMS* and, in the *Event Management* section, choose *Configure Notifications*. In the resulting dialog choose *Configure Destination Paths*.



The destination paths configuration is stored in the destinationPaths.xml file. Changes to this file are picked up on the fly with no need to restart or send an event.

8.2.3. Configuring Event Notifications

To configure notifications for individual events in *OpenNMS Meridian*, navigate to *Configure OpenNMS* and, in the *Event Management section*, choose _Configure Notifications. Then choose Configure Event Notifications.



The event notification configuration is stored in the notifications.xml file. Changes to this file are picked up on the fly with no need to restart or send an event.

8.3. Concepts

Notifications are how *OpenNMS Meridian* informs users about an event that happened in the network, without the users having to log in and look at the UI. The core concepts required to understand notifications are:

- · Events and UEIs
- Users, Groups, and On-Call Roles
- · Duty Schedules
- Destination Paths
- Notification Commands

These concepts fit together to form an *Event Notification Definition*. Also related, but presently only loosely coupled to notifications, are *Alarms* and *Acknowledgments*.

8.3.1. Events and UEIs

As discussed in the chapter on Events, events are central to the operation of *OpenNMS Meridian*. Almost everything that happens in the system is the result of, or the cause of, one or more events; Every notification is triggered by exactly one event. A good understanding of events is therefore essential to a working knowledge of notifications.

Every event has a *UEI* (Uniform Event Identifier), a string uniquely identifying the event's type. UEIs are typically formatted in the style of a URI, but the only requirement is that they start with the string uei. Most notifications are triggered by an exact UEI match (though they may also be triggered with partial UEI matches using regular expression syntax).

8.3.2. Users, Groups, and On-Call Roles

Users are entities with login accounts in the *OpenNMS Meridian* system. Ideally each user corresponds to a person. They are used to control access to the web UI, but also carry contact information (e-mail addresses, phone numbers, etc.) for the people they represent. A user may

receive a notification either individually or as part of a *Group* or *On-Call Role*. Each user has several technology-specific contact fields, which must be filled if the user is to receive notifications by the associated method.

Groups are lists of users. In large systems with many users it is helpful to organize them into *Groups*. A group may receive a notification, which is often a more convenient way to operate than on individual user. *Groups* allow to assign a set of users to *On Call Roles* to build more complex notification workflows.

How to create or modify membership of Users in a Group

- 1. Login as a **User** with administrative permissions
- 2. Choose **Configure OpenNMS** from the user specific main navigation which is named as your login user name
- 3. Choose Configure Users, Groups and On-Call roles and select Configure Groups
- 4. Create a new **Group** with **Add new group** or modify an existing **Group** by clicking the **Modify** icon next to the **Group**
- 5. Select **User** from **Available Users** and use the >> to add them to the **Currently in Group** or select the users in the **Currently in Group** list and use << to remove them from the list.
- 6. Click **Finish** to persist and apply the changes



The order of the *Users* in the group is relevant and is used as the order for *Notifications* when this group is used as *Target* in a *Destination Path*.

How to delete a Group

- 1. Login as a **User** with administrative permissions
- 2. Choose **Configure OpenNMS** from the user specific main navigation which is named as your login user name
- 3. Choose Configure Users, Groups and On-Call roles and select Configure Groups
- 4. Use the trash bin icon next to the *Group* to delete
- 5. Confirm delete request with **OK**

On-Call Roles are an overlay on top of groups, designed to enable OpenNMS Meridian to target the appropriate user or users according to a calendar configuration. A common use case is to have System Engineers in On-Call rotations with a given schedule. The On-Call Roles allow to assign a predefined Duty Schedule to an existing Group with Users. For each On-Call Role a User is assigned as a Supervisor to be responsible for the group of people in this On-Call Role.

How to assign a Group to an On-Call Role

- 1. Login as a **User** with administrative permissions
- 2. Choose **Configure OpenNMS** from the user specific main navigation which is named as your login user name
- 3. Choose Configure Users, Groups and On-Call roles and select Configure On-Call Roles
- 4. Use Add New On-Call Role and set a Name for this On-Call Role, assign an existing Group and

give a meaningful description

- 5. Click **Save** to persist
- 6. Define a *Duty Schedule* in the calendar for the given date by click on the **Plus (+)** icon of the day and provide a notification time for a specific *User* from the associated *Group*
- 7. Click **Save** to persist the *Schedule*
- 8. Click **Done** to apply the changes

8.3.3. Duty Schedules

Every *User* and *Group* may have a *Duty Schedule*, which specifies that user's (or group's) weekly schedule for receiving notifications. If a notification should be delivered to an individual user, but that user is not on duty at the time, the notification will never be delivered to that user. In the case of notifications targeting a user via a group, the logic differs slightly. If the group is on duty at the time the notification is created, then all users who are also on duty will be notified. If the group is on duty, but no member user is currently on duty, then the notification will be queued and sent to the next user who comes on duty. If the group is off duty at the time the notification is created, then the notification will never be sent.

8.3.4. Destination Paths

A *Destination Path* is a named, reusable set of rules for sending notifications. Every destination path has an initial step and zero or more escalation steps.

Each step in a destination path has an associated delay which defaults to zero seconds. The initial step's delay is called the *initial delay*, while an escalation step's delay is simply called its *delay*.

Each step has one or more *targets*. A target may be a user, a group, an on-call role, or a one-off email address.



While it may be tempting to use one-off e-mail addresses any time an individual user is to be targeted, it's a good idea to reserve one-off e-mail addresses for special cases. If a user changes her e-mail address, for instance, you'll need to update in every destination path where it appears. The use of one-off e-mail addresses is meant for situations where a vendor or other external entity is assisting with troubleshooting in the short term.

When a step targets one or more groups, a delay may also be specified for each group. The default is zero seconds, in which case all group members are notified simultaneously. If a longer delay is set, the group members will be notified in alphabetical order of their usernames.



Avoid using the same name for a group and a user. The destination path configuration does not distinguish between users and groups at the step level, so the behavior is undefined if you have both a user and a group named admin. It is for this reason that the default administrators group is called Admin (with a capital A) — case matters.

Within a step, each target is associated with one or more notification commands. If multiple

commands are selected, they will execute simultaneously.

Each step also has an *auto-notify* switch, which may be set to off, on, or auto. This switch specifies the logic used when deciding whether or not to send a notice for an auto-acknowledged notification to a target that was not on duty at the time the notification was first created. If off, notices will never be sent to such a target; if on, they will always be sent; if auto, the system employs heuristics aimed at "doing the right thing".

8.3.5. Notification Commands

A *Notification Command* is a named, reusable execution profile for a Java class or external program command used to convey notices to targets. The following notification commands are included in the default configuration:

callHomePhone, callMobilePhone, and callWorkPhone

Ring one of the phone numbers configured in the user's contact information. All three are implemented using the in-process Asterisk notification strategy, and differ only in which contact field is used.

ircCat

Conveys a notice to an instance of the *IRCcat* Internet Relay Chat bot. Implemented by the inprocess IRCcat notification strategy.

javaEmail and javaPagerEmail

By far the most commonly used commands, these deliver a notice to a user's email or pagerEmail contact field value. By configuring a user's pagerEmail contact field value to target an email-to-SMS gateway, SMS notifications are trivially easy to configure. Both are implemented using the in-process JavaMail notification strategy.

microblogDM, microblogReply, and microblogUpdate

Sends a notice to a user as a direct message, at a user via an at-reply, or to everybody as an update via a microblog service with a Twitter v1-compatible API. Each command is implemented with a separate, in-process notification strategy.

numericPage and textPage

Sends a notice to a user's numeric or alphanumeric pager. Implemented as an external command using the qpage utility.

xmppGroupMessage and xmppMessage

Sends a message to an XMPP group or user. Implemented with the in-process XMPP notification strategy.

Notification commands are customizable and extensible by editing the notificationCommands.xml file.



Use external binary notification commands sparingly to avoid fork-bombing your *OpenNMS Meridian* system. Originally, all notification commands were external. Today only the numericPage and textPage commands use external programs to do their work.

8.4. Bonus Notification Methods

A handful of newer notification methods are included in *OpenNMS Meridian* but currently require manual steps to activate.

8.4.1. Mattermost

If your organization uses the Mattermost team communications platform, you can configure *OpenNMS Meridian* to send notices to any Mattermost channel via an incoming webhook. You must configure an incoming webhook in your Mattermost team and do a bit of manual configuration to your *OpenNMS Meridian* instance.

First, add the following bit of XML to the notificationCommands.xml configuration file (no customization should be needed):

Then create a new file called mattermost.properties in the opennms.properties.d directory with the following contents (customizing values as appropriate):

```
org.opennms.netmgt.notifd.mattermost.webhookURL=https://mattermost.example.com/hooks/bf980352b5f7232efe721dbf0626bee1
```

Restart OpenNMS so that the mattermost.properties file will be loaded. Your new mattermost notification command is now available for use in a destination path.

Additional Options

The following table lists optional properties that you may use in mattermost.properties to customize your Mattermost notifications.



To improve the layout, the property names have been shortened to their final component; you must prepend org.opennms.netmgt.notifd.mattermost. when using them.

Table 77. Additional available parameters for the Mattermost notification strategy

| Param eter | Description | Requi red | Default value | Example |
|---------------|---|--------------|--------------------|---|
| channe 1 | Specify a channel or private group other than the one targeted by the webhook | optio nal | Webhook default | NetOps |
| userna me | The username to associate with the notification posts | optio nal | None | OpenNMS_Bot |
| iconEm oji | An emoji sequence to use as the icon for the notification posts | optio nal | No icon | :metal: |
| iconUR L | The URL of an image to use as the icon for the notification posts | optio nal | No icon | https://example.org/a ssets/icon.png |



Some of the optional configuration parameters are incompatible with some versions of Mattermost. For instance, the channel option is known not to work with Mattermost 3.7.0.

For more information on incoming webhooks in Mattermost, see Mattermost Integration Guide.

8.4.2. Slack Notifications

If your organization uses the Slack team communications platform, you can configure *OpenNMS Meridian* to send notices to any Slack channel via an incoming webhook. You must configure an incoming webhook in your Slack team and do a bit of manual configuration to your *OpenNMS Meridian* instance.

First, add the following bit of XML to the notificationCommands.xml configuration file (no customization should be needed):

Then create a new file called slack.properties in the opennms.properties.d directory with the following contents (customizing values as appropriate):

```
org.opennms.netmgt.notifd.slack.webhookURL=
https://hooks.slack.com/services/AEJ7IIYAI/XOOTH3EOD/c3fc4a662c8e07fe072aeeec
```

Restart OpenNMS so that the slack.properties file will be loaded. Your new slack notification command is now available for use in a destination path.

Additional Options

The following table lists optional properties that you may use in slack.properties to customize your Slack notifications.



To improve the layout, the property names have been shortened to their final component; you must prepend org.opennms.netmgt.notifd.slack. when using them.

Table 78. Additional parameters for the Slack notification strategy

| Param eter | Description | Requi red | Default value | Example |
|---------------|---|--------------|--------------------|---|
| channe l | Specify a channel or private group other than the one targeted by the webhook | optio nal | Webhook default | NetOps |
| userna me | The username to associate with the notification posts | optio nal | None | OpenNMS_Bot |
| iconEm oji | An emoji sequence to use as the icon for the notification posts | optio nal | No icon | :metal: |
| iconUR L | The URL of an image to use as the icon for the notification posts | optio nal | No icon | https://example.org/a ssets/icon.png |

For more information on incoming webhooks in Slack, see Slack API.

Chapter 9. Provisioning

9.1. Introduction

The introduction of OpenNMS version 1.8 empowers enterprises and services providers like never before with a new service daemon for maintaining the managed entity inventory in OpenNMS. This new daemon, *Provisiond*, unifies all previous entity control mechanisms available in 1.6 (*Capsd* and the *Importer*), into a new and improved, massively parallel, policy based provisioning system. System integrators should note, *Provisiond* comes complete with a *RESTFul Web Service API* for easy integration with external systems such as CRM or external inventory systems as well as an adapter API for interfacing with other management systems such as configuration management.

OpenNMS 1.0, introduced almost a decade ago now, provided a capabilities scanning daemon, Capsd, as the mechanism for provisioning managed entities. *Capsd*, deprecated with the release of 1.8.0, provided a rich automatic provisioning mechanism that simply required an IP address to seed its algorithm for creating and maintaining the managed entities (nodes, interfaces, and IP based services). Version 1.2 added and *XML-RPC API* as a more controlled (directed) strategy for provisioning services that was mainly used by non telco based service providers (i.e. managed hosting companies). Version 1.6 followed this up with yet another and more advanced mechanism called the *Importer service daemon*. The *Importer* provided large service providers with the ability to strictly control the OpenNMS entity provisioning with an XML based API for completely defining and controlling the entities where no discovery and service scanning scanning was feasible.

The Importer service improved OpenNMS' scalability for maintaining managed entity databases by an order of magnitude. This daemon, while very simple in concept and yet extremely powerful and flexible provisioning improvement, has blazed the trail for Provisiond. The Importer service has been in production for 3 years in service provider networks maintaining entity counts of more than 50,000 node level entities on a single instances of OpenNMS. It is a rock solid provisioning tool.

Provisiond begins a new era of managed entity provisioning in OpenNMS.

9.2. Concepts

Provisioning is a term that is familiar to service providers (a.k.a. operators, a.k.a. telephone companies) and OSS systems but not so much in the non OSS enterprises.

Provisiond receives "requests" for adding managed entities via 2 basic mechanisms, the OpenNMS Meridian traditional "New Suspect" event, typically via the *Discovery daemon*, and the import requisition (XML definition of node entities) typically via the Provisioning Groups UI. If you are familiar with all previous releases of OpenNMS, you will recognize the *New Suspect Event* based *Discovery* to be what was previously the *Capsd* component of the auto discovery behavior. You will also recognize the import requisition to be of the *Model Importer* component of OpenNMS. *Provisiond* now unifies these two separate components into a massively parallel advanced policy based provisioning service.

9.2.1. Terminology

The following terms are used with respect to the OpenNMS Meridian provisioning system and are essential for understanding the material presented in this guide.

Entity

Entities are managed objects in OpenNMS Meridian such as Nodes, IP interfaces, SNMP Interfaces, and Services.

Foreign Source and Foreign ID

The *Importer* service from 1.6 introduced the idea of foreign sources and foreign IDs. The *Foreign Source* uniquely identifies a provisioning source and is still a basic attribute of importing node entities into OpenNMS Meridian. The concept is to provide an external (foreign) system with a way to uniquely identify itself and any node entities that it is requesting (via a requisition) to be provisioned into OpenNMS Meridian.

The *Foreign ID* is the unique node ID maintained in foreign system and the foreign source uniquely identifies the external system in OpenNMS Meridian.

OpenNMS Meridian uses the combination of the foreign source and foreign ID become the unique foreign key when synchronizing the set of nodes from each source with the nodes in the OpenNMS Meridian DB. This way the foreign system doesn't have to keep track of the OpenNMS Meridian node IDs that are assigned when a node is first created. This is how *Provisiond* can decided if a node entity from an import requisition is new, has been changed, or needs to be deleted.

Foreign Source Definition

Additionally, the foreign source has been extended to also contain specifications for how entities should be discovered and managed on the nodes from each foreign source. The name of the foreign source has become pervasive within the provisioning system and is used to simply some of the complexities by weaving this name into:

- the name of the provisioning group in the Web-UI
- the name of the file containing the persisted requisition (as well as the pending requisition if it is in this state)
- the foreign-source attribute value inside the requisition (obviously, but, this is pointed out to indicate that the file name doesn't necessarily have to equal the value of this attribute but is highly recommended as an OpenNMS Meridian best practice)
- the building attribute of the node defined in the requisition (this value is called "site" in the Web-UI and is assigned to the building column of the node's asset record by Provisiond and is the default value used in the Site Status View feature)

Import Requisition

Import requisition is the terminology OpenNMS Meridian uses to represent the set of nodes, specified in XML, to be provisioned from a foreign source into OpenNMS Meridian. The requisition schema (XSD) can be found at the following location. http://xmlns.opennms.org/xsd/config/model-

import

Auto Discovery

Auto discovery is the term used by OpenNMS Meridian to characterize the automatic provisioning of nodes entities. Currently, OpenNMS Meridian uses an ICMP ping sweep to find IP address on the network. For the IPs that respond and that are not currently in the DB, OpenNMS Meridian generates a new suspect event. When this event is received by Provisiond, it creates a node and it begins a node scan based on the default foreign source definition.

Directed Discovery

Provisiond takes over for the Model Importer found in version 1.6 which implemented a unique, first of its kind, controlled mechanism for specifying managed entities directly into OpenNMS Meridian from one or more data sources. These data sources often were in the form of an inhoused developed inventory or stand-alone provisioning system or even a set of element management systems. Using this mechanism, OpenNMS Meridian is directed to add, update, or delete a node entity exactly as defined by the external source. No discovery process is used for finding more interfaces or services.

Enhanced Directed Discovery

Directed discovery is enhanced with the capability to scan nodes that have been directed nodes for entities (interfaces.

Policy Based Discovery

The phrase, Policy based Directed Discovery, is a term that represents the latest step in OpenNMS Meridian provisioning evolution and best describes the new provisioning architecture now in OpenNMS Meridian for maintaining its inventory of managed entities. This term describes the control that is given over the Provisioning system to OpenNMS Meridian users for managing the behavior of the NMS with respect to the new entities that are being discovered. Current behaviors include persistence, data collection, service monitoring, and categorization policies.

9.2.2. Addressing Scalability

The explosive growth and density of the IT systems being deployed today to support not traditional IP services is impacting management systems like never before and is demanding from them tremendous amounts of scalability. The scalability of a management system is defined by its capacity for maintaining large numbers of managing entities coupled with its efficiency of managing the entities.

Today, It is not uncommon for OpenNMS Meridian deployments to find node entities with tens of thousands of physical interfaces being reported by SNMP agents due to virtualization (virtual hosts, interfaces, as well as networks). An NMS must be capable of using the full capacity every resource of its computing platform (hardware and OS) as effectively as possible in order to manage these environments. The days of writing scripts or single threaded applications will just no longer be able to do the work required an NMS when dealing with the scalability challenges facing systems and systems administrators working in this domain.

Parallelization and Non-Blocking I/O

Squeezing out every ounce of power from a management system's platform (hardware and OS) is absolutely required to complete all the work of a fully functional NMS such as OpenNMS Meridian. Fortunately, the hardware and CPU architecture of a modern computing platform provides multiple CPUs with multiple cores having instruction sets that include support for atomic operations. While these very powerful resources are being provided by commodity systems, it makes the complexity of developing applications to use them vs. not using them, orders of magnitude more complex. However, because of scalability demands of our complex IT environments, multi-threaded NMS applications are now essential and this has fully exposed the complex issues of concurrency in software development.

OpenNMS Meridian has stepped up to this challenge with its new concurrency strategy. This strategy is based on a technique that combines the efficiency of parallel (asynchronous) operations (traditionally used by most effectively by single threaded applications) with the power of a fully current, non-blocking, multi-threaded design. The non-blocking component of this new concurrency strategy added greater complexity but OpenNMS Meridian gained orders of magnitude in increased scalability.



Java Runtimes, based on the Sun JVM, have provided implementations for processor based atomic operations and is the basis for OpenNMS Meridian' non-blocking concurrency algorithms.

Provisioning Policies

Just because you can, doesn't mean you should! Because the massively parallel operations being created for *Provisiond* allows tremendous numbers of nodes, interfaces, and services to be very rapidly discovered and persisted, doesn't mean it should. A *policy API* was created for *Provisiond* that allows implementations to be developed that can be applied to control the behavior of *Provisiond*. The 1.8 release includes a set of flexible provisioning policies that control the persistence of entities and their attributes constrain monitoring behavior.

When nodes are imported or re-scanned, there is, potentially, a set of zero or more provisioning policies that are applied. The policies are defined in the foreign source's definition. The policies for an auto-discovered node or nodes from provisioning groups that don't have a foreign source definition, are the policies defined in the default foreign source definition.

The Default Foreign Source Definition

Contained in the libraries of the Provisioning service is the "template" or default foreign source. The template stored in the library is used until the OpenNMS Meridian admin user alters the default from the *Provisioning Groups* WebUI. Upon edit, this template is exported to the OpenNMS Meridian etc/ directory with the file name: default-foreign-source.xml.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<foreign-source date-stamp="2009-10-16T18:04:12.844-05:00"</pre>
                 name="default"
                xmlns=
"http://xmlns.opennms.org/[http://xmlns.opennms.org/xsd/config/foreign-source">
    <scan-interval>1d</scan-interval>
    <detectors>
      <detector class="org.opennms.netmgt.provision.detector.datagram.DnsDetector"</pre>
name="DNS"/>
      <detector class="org.opennms.netmgt.provision.detector.simple.FtpDetector" name</pre>
="FTP"/>
      <detector class="org.opennms.netmgt.provision.detector.simple.HttpDetector"</pre>
name="HTTP"/>
      <detector class="org.opennms.netmgt.provision.detector.simple.HttpsDetector"</pre>
name="HTTPS"/>
      <detector class="org.opennms.netmgt.provision.detector.icmp.IcmpDetector" name=</pre>
"ICMP"/>
      <detector class="org.opennms.netmgt.provision.detector.simple.ImapDetector"</pre>
name="IMAP"/>
      <detector class="org.opennms.netmgt.provision.detector.simple.LdapDetector"</pre>
name="LDAP"/>
      <detector class="org.opennms.netmgt.provision.detector.simple.NrpeDetector"</pre>
      <detector class="org.opennms.netmgt.provision.detector.simple.Pop3Detector"</pre>
name="POP3"/>
      <detector class="
org.opennms.netmgt.provision.detector.radius.RadiusAuthDetector" name="Radius"/>
      <detector class="org.opennms.netmgt.provision.detector.simple.SmtpDetector"</pre>
name="SMTP"/>
      <detector class="org.opennms.netmgt.provision.detector.snmp.SnmpDetector" name=</pre>
"SNMP"/>
      <detector class="org.opennms.netmgt.provision.detector.ssh.SshDetector" name=</pre>
"SSH"/>
 </detectors>
 <policies/>
</foreign-source>
```

Automatic Rescanning

The default foreign source defines a scan-interval of 1d, which will cause all nodes in the requisition to be scanned daily. You may set the scan interval using any combination of the following signifiers:

- · w: Weeks
- d: Days
- h: Hours
- m: Minutes
- s: Seconds

• ms: Milliseconds

For example, to rescan every 6 days and 53 minutes, you would set the scan-interval to 6d 53m.

Don't forget, for the new scan interval to take effect, you will need to import the requisition one more time so that the foreign source becomes active.

Disabling Rescan

For a large number of devices, you may want to set the scan-interval to 0 to disable automatic rescan altogether. OpenNMS Meridian will not attempt to rescan the nodes in the requisition unless you trigger a manual (forced) rescan through the web UI or Provisioning ReST API.

9.3. Getting Started

An NMS is of no use until it is setup for monitoring and entities are added to the system. OpenNMS Meridian installs with a base configuration with a configuration that is sufficient get service level monitoring and performance management quickly up and running. As soon as managed entities are provisioned, the base configuration will automatically begin monitoring and reporting.

Generally speaking, there are two methods of provisioning in OpenNMS Meridian: *Auto Discovery* and *Directed Discovery*. We'll start with *Auto Discovery*, but first, we should quickly review the configuration of SNMP so that newly discovered devices can be immediately scanned for entities as well as have reporting and thresholding available.

9.3.1. Provisioning the SNMP Configuration

OpenNMS Meridian requires SNMP configuration to be properly setup for your network in order to properly understand Network and Node topology as well as to automatically enable performance data collection. Network topology is updated as nodes (a.k.a. devices or hosts) are provisioned. Navigate to the *Admin/Configure SNMP Community Names by IP address* as shown below.

Configuring SNMP community names

Descriptions

SNMP Config Lookup: You can look up the actual SNMP configuration for a spectox and press "Look up". The configuration will then be shown in the "Updateing

Updating SNMP Configuration: In the boxes on the left, enter in a specific IP add community string, and other SNMP parameters.

OpenNMS will optimize this list, so enter the most generic first (i.e. the largest ra added that includes a specific IP address, the community name for the specific s

For devices that have already been provisioned and that have an event stating th changed, it may be necessary to update the SNMP information on the interface pathese changes to take effect.

| v1/v2c specific parameters | |
|----------------------------|------------------|
| Read Community String: | YrusoNoz |
| | Default: public |
| Write Community String: | |
| | Default: private |
| | |

Sta

| Descriptions | | |
|---|--------------------|--|
| | | ip the actual SNMP configuration for a spe iration will then be shown in the "Updatein |
| Updating SNMP Co community string, a | | boxes on the left, enter in a specific IP ad arameters. |
| | | nter the most generic first (i.e. the largest r ress, the community name for the specific |
| | e necessary to upd | rovisioned and that have an event stating t ate the SNMP information on the interface |
| v1/v2c specific par | rameters | |
| | nunity String: | YrusoNoz |
| | | Default: public |
| Write Comm | nunity String: | |
| | | Default: private |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Descriptions

SNMP Config Lookup: You can look up the actual SNMP configuration for a spectox and press "Look up". The configuration will then be shown in the "Updateing

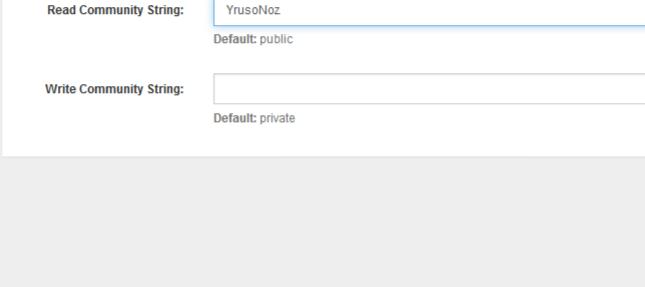
Updating SNMP Configuration: In the boxes on the left, enter in a specific IP add community string, and other SNMP parameters.

OpenNMS will optimize this list, so enter the most generic first (i.e. the largest ra added that includes a specific IP address, the community name for the specific s

For devices that have already been provisioned and that have an event stating th changed, it may be necessary to update the SNMP information on the interface pathese changes to take effect.

| v1/v2c specific parameters | |
|----------------------------|------------------|
| Read Community String: | YrusoNoz |
| | Default: public |
| Write Community String: | |
| | Default: private |
| | |

Descriptions SNMP Config Lookup: You can look up the actual SNMP configuration for a specific IP. To do so enter the IP Address in box and press "Look up". The configuration will then be shown in the "Updateing SNMP Community Names" area. Updating SNMP Configuration: In the boxes on the left, enter in a specific IP address and community string, or a range community string, and other SNMP parameters. OpenNMS will optimize this list, so enter the most generic first (i.e. the largest range) and the specific IP addresses la added that includes a specific IP address, the community name for the specific address will be changed to be that of the specific address and community specific address and community specific IP address. V1/V2c specific parameters YrusoNoz Default: public





Provisiond includes an option to add community information in the *Single Node* provisioning interface. This, is equivalent of entering a single IP address in the screen with the convenience of setting the community string at the same time a node is provisioned. See the *Quick Node Add* feature below for more details about this capability.

This screen sets up SNMP within OpenNMS Meridian for agents listening on IP addresses 10.1.1.1 through 10.254.254.254. These settings are optimized into the snmp-configuration.xml file. Optimization means that the minimal configuration possible will be written. Any IP addresses already configured that are eclipsed by this range will be removed. Here is the resulting configuration.

```
<?xml version="1.0" encoding="UTF-8"?>

<snmp-config
xmlns="http://xmlns.opennms.org/xsd/config/snmp[http://xmlns.opennms.org/xsd/config/sn
mp]"
port="161" retry="3" timeout="800" read-community="public"

version="v1" max-vars-per-pdu="10">

<definition retry="1" timeout="2000"

read-community="public" version="v2c">

<specific>10.12.23.32</specific>

</definition>

</snmp-config>
```

However, If an IP address is then configured that is within the range, the range will be split into two separate ranges and a specific entry will be added. For example, if a configuration was added through the same UI for the IP: 10.12.23.32 having the community name public, then the resulting configuration will be:

```
<?xml version="1.0" encoding="UTF-8"?>
<snmp-config xmlns="http://xmlns.opennms.org/xsd/config/snmp"</pre>
             port="161"
             retry="3"
             timeout="800"
             read-community="public"
             version="v1"
             max-vars-per-pdu="10">
    <definition retry="1" timeout="2000" read-community="YrusoNoz" version="v2c">
        <range begin="10.1.1.1" end="10.12.23.31"/>
        <range begin="10.12.23.33" end="10.254.254.254"/>
    </definition>
    <definition retry="1" timeout="2000" read-community="public" version="v2c">
        <specific>10.12.23.32</specific>
    </definition>
</snmp-config>
```



the bold IP addresses show where the range was split and the specific with community name "public" was added.

Now, with SNMP configuration provisioned for our 10 networks, we are ready to begin adding nodes. Our first example will be to automatically discover and add all managed entities (nodes, IP interfaces, SNMP Interfaces, and Monitored IP based Services). We will then give an example of how to be more *directed* and deliberate about your discovery by using *Provisioning Groups*.

Automatically discovered entities are analyzed, persisted to the relational data store, and then managed based on the policies defined in the default foreign source definition. This is very similar to the way that entities were previously handled by the (now obsolete) Capsd daemon but with finer grained sense of control.

9.3.2. Automatic Discovery

Currently in OpenNMS Meridian, the ICMP is used to automatically provision node entities into OpenNMS Meridian. This functionality has been in OpenNMS since is 1.0 release, however, in 1.8, a few of the use cases have been updated with *Provisiond's* replacement of *Capsd*.

Separation of Concerns

Version 1.8 *Provisiond* separates what was called *Capsd* scanning in to 3 distinct phases: entity scanning, service detection, and node merging. These phases are now managed separately by Provisiond. Immediately following the import of a node entity, tasks are created for scanning a node to discover the node entity's interfaces (SNMP and IP). As interfaces are found, they are persisted and tasks are scheduled for service detection of each IP interface.

For auto discovered nodes, a node merging phase is scheduled; Nodes that have been directly provisioned will not be included in the node merging process. Merging will only occur when 2 automatically discovered nodes appear to be the same node.



the use case and redesign of node merging is still an outstanding issue with the 1.8.0 release

9.3.3. Enhanced Directed Discovery

This new form of provisioning first appears in OpenNMS with version 1.8 and the new Provisiond service. It combines the benefits of the Importer's strictly controlled methodology of directed provisioning (from version 1.6) with OpenNMS' robustly flexible auto discovery. *Enhanced Directed discovery* begins with an enhanced version of the same import requisition used in directed provisioning and completes with a policy influenced persistence phase that sorts though the details of all the entities and services found during the entity and service scanning phase.

If you are planning to use this form of provisioning, it important to understand the conceptual details of how *Provisiond* manages entities it is *directed* to provision. This knowledge will enable administrators and systems integrators to better plan, implement, and resolve any issues involved with this provisioning strategy.

Understanding the Process

There are 3 phases involved with directing entities to be discovered: import, node scan, and service scan. The import phase also has sub phases: marshal, audit, limited SNMP scan, and re-parent.

Marshal and Audit Phases

It is important to understand that the nodes requisitioned from each foreign source are managed as a complete set. Nodes defined in a requisition from the foreign source *CRM* and *CMDB*, for example, will be managed separately from each other even if they should contain exactly the same node definitions. To OpenNMS Meridian, these are individual entities and they are managed as a set.

Requisitions are referenced via a URL. Currently, the URL can be specified as one of the following protocols: FILE, HTTP, HTTPS, and DNS. Each protocol has a protocol handler that is used to stream the XML from a *foreign source*, i.e. http://inv.corp.org/import.cgi?customer=acme or file:/opt/opennms/etc/imports/acme.xml. The DNS protocol is a special handler developed for Provisioning sets of nodes as a *foreign-source* from a corporate DNS server. See DNS Protocol Handler for details.

Upon the import request (either on schedule or on demand via an Event) the requisition is marshaled into Java objects for processing. The nodes defined in the requisition represent what OpenNMS Meridian should have as the current set of managed entities from that foreign source. The audit phase determines for each node defined (or not defined) in the requisition which are to be processed as an *Add*, *Update*, or *Delete* operation during the *Import Phase*. This determination is made by comparing the set foreign IDs of each node in the requisition set with the set of foreign IDs of currently managed entities in OpenNMS Meridian.

The intersection of the IDs from each set will become the Update operations, the extra set of foreign IDs that are in the requisition become the Add operations, and the extra set of foreign IDs from the managed entities become the Delete operations. This implies that the foreign IDs from each foreign source must be unique.

Naturally, the first time an import request is processed from a foreign source there will be zero (0) node entities from the set of nodes currently being managed and each node defined in the requisition will become an Add Operation. If a requisition is processed with zero (0) node definitions, all the currently managed nodes from that foreign source will become Delete operations (all the nodes, interfaces, outages, alarms, etc. will be removed from OpenNMS Meridian).

When nodes are provisioned using the Provisioning Groups Web-UI, the requisitions are stored on the local file system and the file protocol handler is used to reference the requisition. Each Provisioning Group is a separate foreign source and unique foreign IDs are generated by the Web-UI. An MSP might use Provisioning Groups to define the set of nodes to be managed by customer name where each customer's set of nodes are maintained in a separate Provisioning Group.

Import Phase

The import phase begins when Provisiond receives a request to import a requisition from a URL. The first step in this phase is to load the requisition and marshal all the node entities defined in the requisition into Java objects.

If any syntactical or XML structural problems occur in the requisition, the entire import is abandoned and no import operations are completed.

Once the requisition is marshaled, the requisition nodes are audited against the persisted node

entities. The set of requisitioned nodes are compared with a subset of persisted nodes and this subset is generated from a database query using the foreign source defined in the requisition. The audit generates one of three operations for each requisition node: *insert*, *update*, *delete* based on each requisitioned node's foreign ID. Delete operations are created for any nodes that are not in the requisition but are in the DB subset, update operations are created for requisition nodes that match a persisted node from the subset (the intersection), and insert operations are created from the remaining requisition nodes (nodes in the requisition that are not in the DB subset).

If a requisition node has an interface defined as the Primary SNMP interface, then during the update and insert operations the node will be scanned for minimal SNMP attribute information. This scan find the required node and SNMP interface details required for complete SNMP support of the node and only the IP interfaces defined in the requisition.



this not the same as Provisiond SNMP discovery scan phases: node scan and interface scan.

Node Scan Phase

Where directed discovery leaves off and enhanced directed discovery begins is that after all the operations have completed, directed discovery is finished and enhanced directed discovery takes off. The requisitioned nodes are scheduled for node scans where details about the node are discovered and interfaces that were not directly provisioned are also discovered. All physical (SNMP) and logical (IP) interfaces are discovered and persisted based on any *Provisioning Policies* that may have been defined for the foreign source associated with the import requisition.

Service Scan (detection) Phase

Additionally, the new Provisiond enhanced directed discovery mechanism follows interface discovery with service detection on each IP interface entity. This is very similar to the Capsd plugin scanning found in all former releases of OpenNMS except that the foreign source definition is used to define what services should be detected on these interfaces found for nodes in the import requisition.

9.4. Import Handlers

9.4.1. File Handler

9.4.2. HTTP Handler

9.4.3. DNS Handler

The new Provisioning service in OpenNMS Meridian is continuously improving and adapting to the needs of the community.

One of the most recent enhancements to the system is built upon the very flexible and extensible API of referencing an import requisition's location via a URL. Most commonly, these URLs are files on the file system (i.e. file:/opt/opennms/etc/imports/<my-provisioning-group.xml>) as requisitions created by the Provisioning Groups UI. However, these same requisitions for adding, updating, and

deleting nodes (based on the original model importer) can also come from URLs specifying the HTTP protocol: http://myinventory.server.org/nodes.cgi

Now, using Java's extensible protocol handling specification, a new protocol handler was created so that a URL can be specified for requesting a *Zone Transfer (AXFR) request* from a DNS server. The A records are recorded and used to build an import requisition. This is handy for organizations that use DNS (possibly coupled with an IP management tool) as the data base of record for nodes in the network. So, rather than ping sweeping the network or entering the nodes manually into OpenNMS Meridian Provisioning UI, nodes can be managed via 1 or more DNS servers.

The format of the URL for this new protocol handler is: dns://<host>[:port]/<zone>[/<foreign-source>/][?expression=<regex>]

DNS Import Examples:

Simple

dns://my-dns-server/myzone.com

This URL will import all A records from the host my-dns-server on port 53 (default port) from zone "myzone.com" and since the foreign source (a.k.a. the provisioning group) is not specified it will default to the specified zone.

Using a Regular Expression Filter

dns://my-dns-server/myzone.com/portland/?expression=^por-.*

This URL will import all nodes from the same server and zone but will only manage the nodes in the zone matching the regular expression 'port-.* and will and they will be assigned a unique foreign source (provisioning group) for managing these nodes as a subset of nodes from within the specified zone.

If your expression requires URL encoding (for example you need to use a ? in the expression) it must be properly encoded.

dns://my-dns-server/myzone.com/portland/?expression=^por[0-9]%3F

DNS Setup

Currently, the DNS server requires to be setup to allow a zone transfer from the OpenNMS Meridian server. It is recommended that a secondary DNS server is running on OpenNMS Meridian and that the OpenNMS Meridian server be allowed to request a zone transfer. A quick way to test if zone transfers are working is:

dig -t AXFR @<dnsServer> <zone>

Configuration

The configuration of the Provisoning system has moved from a properties file (model-importer.properties) to an XML based configuration container. The configuration is now extensible to allow the definition of 0 or more import requisitions each with their own cron based schedule for automatic importing from various sources (intended for integration with external URL such as http and this new dns protocol handler.

A default configuration is provided in the OpenNMS Meridian etc/ directory and is called: provisiond-configuration.xml. This default configuration has an example for scheduling an import from a DNS server running on the localhost requesting nodes from the zone, localhost and will be imported once per day at the stroke of midnight. Not very practical but is a good example.

```
<?xml version="1.0" encoding="UTF-8"?>
   xsi:schemaLocation="http://xmlns.opennms.org/xsd/config/provisiond-configuration"
                           foreign-source-dir="/opt/opennms/etc/foreign-sources"
                           requistion-dir="/opt/opennms/etc/imports"
                           importThreads="8"
                           scanThreads="10"
                           rescanThreads="10"
                           writeThreads="8" >
   <!--http://www.quartz-scheduler.org/documentation/quartz-1.x/tutorials/crontrigger
       Field Name Allowed Values Allowed Special Characters
       Seconds 0-59 , - * / Minutes 0-59 , - * / Hours 0-23 , - * /
       Day-of-month1-31, - * ? / L W C Month1-12 or JAN-DEC, - * /
       Day-of-Week1-7 or SUN-SAT, - * ? / L C # Year (Opt)empty, 1970-2099, - * /
   -->
   <requisition-def import-name="localhost"
                   import-url-resource="dns://localhost/localhost">
       <cron-schedule>0 0 0 * * ? *</cron-schedule> <!-- daily, at midnight -->
   </requisition-def>
</provisiond-configuration>
```

Configuration Reload

Like many of the daemon configuration in the 1.7 branch, the configurations are reloadable without having to restart OpenNMS Meridian, using the reloadDaemonConfig uei:

```
/opt/opennms/bin/send-event.pl
uei.opennms.org/internal/reloadDaemonConfig --parm 'daemonName Provisiond'
```

This means that you don't have to restart OpenNMS Meridian every time you update the configuration.

9.5. Provisioning Examples

Here are a few practical examples of enhanced directed discovery to help with your understanding of this feature.

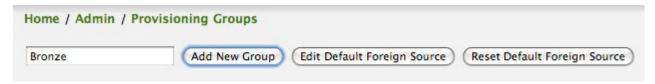
9.5.1. Basic Provisioning

This example adds three nodes and requires no OpenNMS Meridian configuration other than specifying the node entities to be provisioned and managed in OpenNMS Meridian.

Defining the Nodes via the Web-UI

Using the Provisioning Groups Web-UI, three nodes are created given a single IP address. Navigate to the Admin Menu and click Provisioning Groups Menu from the list of Admin options and create the group *Bronze*.

Creating a new Provisioning Group



Clicking the *Add New Group* button will create the group and will redisplay the page including this new group among the list of any group(s) that have already been created.

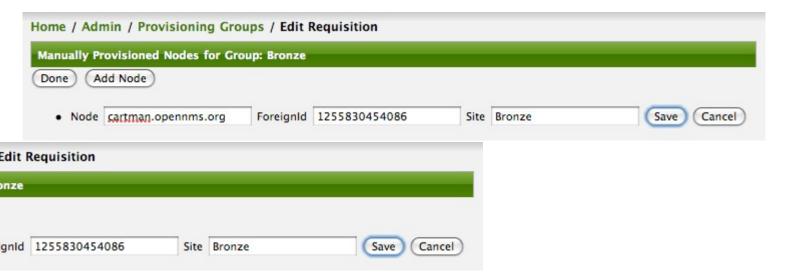




At this point, the XML structure for holding the new provisioning group (a.k.a. an import requisition) has been persisted to the '\$OPENNMS_ETC/imports/pending' directory.

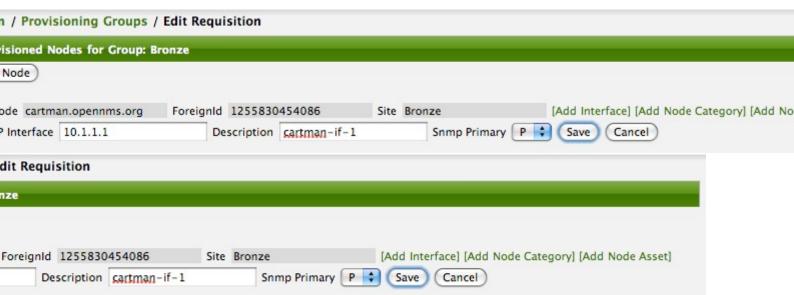
Clicking the *Edit* link will bring you to the screen where you can begin the process of defining node entities that will be imported into OpenNMS Meridian. Click the Add Node button will begin the node entity creation process fill in the node label and click the *Save* button.

Creating a new Node definition in the Provisioning Group



At this point, the provisioning group contains the basic structure of a node entity but it is not complete until the interface(s) and interface service(s) have been defined. After having clicked the *Save* button, as we did above presents, in the Web-UI, the options *Add Interface*, *Add Node Category*, and *Add Node Asset*. Click the *Add Interface* link to add an interface entity to the node.

Adding an Interface to the node definition



Enter the IP address for this interface entity, a description, and specify the Primary attribute as P (Primary), S (Secondary), N (Not collected), or C (Collected) and click the save button. Now the node entity has an interface for which services can be defined for which the Web-UI now presents the *Add Service* link. Add two services (ICMP, SNMP) via this link.

A complete node definition with all required elements defined.



Now the node entity definition contains all the *required* elements necessary for importing this requisition into OpenNMS Meridian. At this point, all the interfaces that are required for the node should be added. For example, NAT interfaces should be specified there are services that they provide because they will not be discovered during the Scan Phase.

Two more node definitions will be added for the benefit of this example.

The completed requisition for the example Bronze Provisioning Group



■ Service SNMP

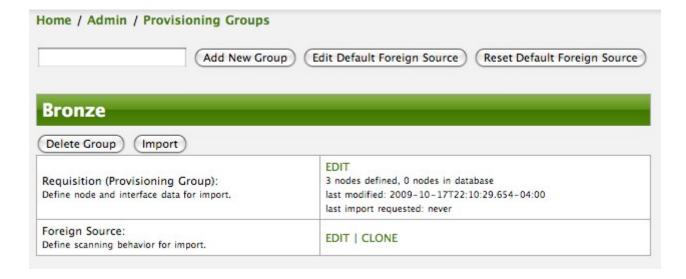
| dit Requi | sition | | | | | | | |
|-----------|--------------|-------------|------|--------|--------------|---|---------------|--|
| nze | | | | | | | | |
| | | | | | | | | |
| ForeignId | 125583174300 | 7 | Site | Bronze | | | [Add Interfa | ce] [Add Node Category] [Add Node Asset] |
| | Description | timmy-if-1 | | | Snmp Primary | P | | Add Service |
| | | | | | | | | |
| ForeignId | 125583169651 | .6 | Site | Bronze | | | [Add Interfa | ce] [Add Node Category] [Add Node Asset] |
| | Description | barbrady-if | -1 | | Snmp Primary | P | | Add Service |
| | | | | | | | | |
| | | | | | | | | |
| ForeignId | 125583045408 | 6 | Site | Bronze | | | [Add Interfa- | ce] [Add Node Category] [Add Node Asset] |
| | Description | cartman-if- | -1 | | Snmp Primary | P | | Add Service |
| | | | | | | | | |
| | | | | | | | | |

This set of nodes represents an import requisition for the *Bronze* provisioning group. As this requisition is being edited via the WebUI, changes are being persisted into the OpenNMS Meridian configuration directory '\$OPENNMS_etc/imports/' pending as an XML file having the name bronze.xml.



The name of the XML file containing the import requisition is the same as the provisioning group name. Therefore naming your provisioning group without the use of spaces makes them easier to manage on the file system.

Click the *Done* button to return to the *Provisioning Groups* list screen. The details of the "Bronze" group now indicates that there are 3 nodes in the requisition and that there are no nodes in the DB from this group (a.k.a. foreign source). Additionally, you can see that time the requisition was last modified and the time it last imported are given (the time stamps are stored as attributes inside the requisition and are not the file system time stamps). These details are indicative of how well the DB represents what is in the requisition.







You can tell that this is a pending requisition for 2 reasons: 1) there are 3 nodes defined and 0 nodes in the DB, 2) the requisition has been modified since the last import (in this case *never*).

Import the Nodes

In this example, you see that there are 3 nodes in the pending requisition and 0 in the DB. Click the *Import* button to submit the requisition to the provisioning system (what actually happens is that the Web-UI sends an event to the Provisioner telling it to begin the Import Phase for this group).



Do not refresh this page to check the values of these details. To refresh the details to verify the import, click the *Provisioning Groups* bread crumb item.

You should be able to immediately verify the importation of this provisioning group because the import happens very quickly. Provisiond has several threads ready for processing the import operations of the nodes defined in this requisition.

A few SNMP packets are sent and received to get the SNMP details of the node and the interfaces defined in the requisition. Upon receipt of these packets (or not) each node is inserted as a DB transaction.

The nodes are now added to OpenNMS Meridian and are under management.



Following the import of a node with thousands of interfaces, you will be able to refresh the Interface table browser on the Node page and see that interfaces and services are being discovered and added in the background. This is the discovery component of directed discovery.

Adding a Node

To direct that another node be added from a foreign source (in this example the Bronze Provisioning Group) simply add a new node definition and re-import. It is important to remember that all the node definitions will be re-imported and the existing managed nodes will be updated, if necessary.

Changing a Node

To direct changes to an existing node, simply add, change, or delete elements or attributes of the node definition and re-import. This is a great feature of having directed specific elements of a node in the requisition because that attributes will simply be changed. For example, to change the IP address of the Primary SNMP interface for the node, *barbrady.opennms.org*, just change the requisition and re-import.

Each element in the Web-UI has an associated Edit icon Click this icon to change the IP address for barbrady.opennms.org, click save, and then Click the Done button.

Changing the IP address of barbrady.opennms.org from 10.1.1.2 to 192.168.1.1



The Web-UI will return you to the *Provisioning Groups* screen where you will see that there are the time stamp showing that the requisition's last modification is more recent that the last import time.

The Provisioning Group must be re-imported



This provides an indication that the group must be re-imported for the changes made to the requisition to take effect. The IP Interface will be simply updated and all the required events (messages) will be sent to communicate this change within OpenNMS Meridian.

The IP interface for barbrady.opennms.org is immediately updated



Deleting a Node

Barbrady has not been behaving, as one might expect, so it is time to remove him from the system. Edit the provisioning group, click the delete button next to the node *barbrady.opennms.org*, click the *Done* button.

Bronze Provisioning Group definition indicates a node has been removed and requires an import to delete the node entity from the OpenNMS Meridian system



Click the Import button for the Bronze group and the Barbrady node and its interfaces, services, and any other related data will be immediately deleted from the OpenNMS Meridian system. All the required Events (messages) will be sent by Provisiond to provide indication to the OpenNMS Meridian system that the node Barbrady has been deleted.

Barbrady has been deleted



Deleting all the Nodes

There is a convenient way to delete all the nodes that have been provided from a specific foreign source. From the main *Admin/Provisioning Groups* screen in the Web-UI, click the *Delete Nodes* button. This button deletes all the nodes defined in the Bronze requisition. It is very important to note that once this is done, it cannot be undone! Well it can't be undone from the Web-UI and can only be undone if you've been good about keeping a backup copy of your '\$OPENMS_ETC/' directory tree. If you've made a mistake, before you re-import the requisition, restore the Bronze.xml requisition from your backup copy to the '\$OPENNMS_ETC/imports' directory.

All node definitions have been removed from the Bronze requisition. The Web-UI indicates an import is

now required to remove them from OpenNMS Meridian.



Clicking the *Import* button will cause the *Audit Phase* of *Provisiond* to determine that all the nodes from the *Bronze* group (foreign source) should be deleted from the DB and will create *Delete* operations. At this point, if you are satisfied that the nodes have been deleted and that you will no longer require nodes to be defined in this Group, you will see that the *Delete Nodes* button has now changed to the *Delete Group* button. The *Delete Group* button is displayed when there are no nodes entities from that group (foreign source) in OpenNMS Meridian.

When no node entities from the group exist in OpenNMS Meridian, then the *Delete Group* button is displayed.

9.5.2. Advanced Provisioning Example

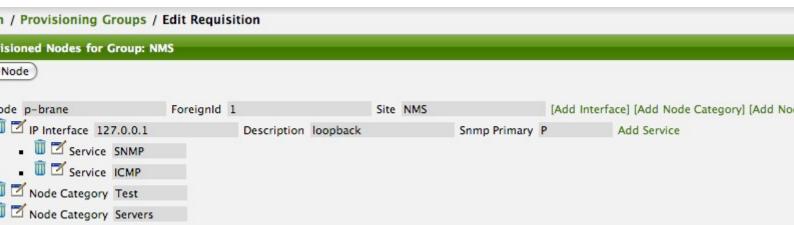
In the previous example, we provisioned 3 nodes and let *Provisiond* complete all of its import phases using a default foreign source definition. Each Provisioning Group can have a separate foreign source definition that controls:

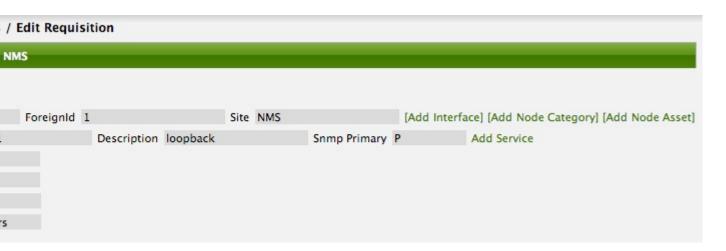
- The rescan interval
- · The services to be detected
- The policies to be applied

This example will demonstrate how to create a foreign source definition and how it is used to control the behavior of Provisiond when importing a *Provisioning Group/foreign source requisition*.

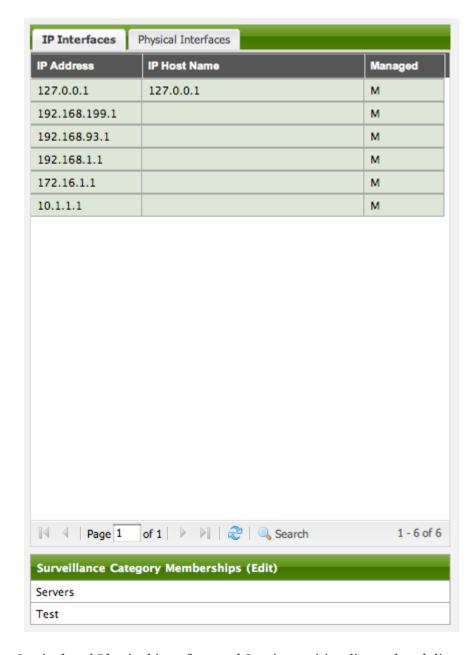
First let's simply provision the node and let the default foreign source definition apply.

The node definition used for the Advanced Provisioning Example

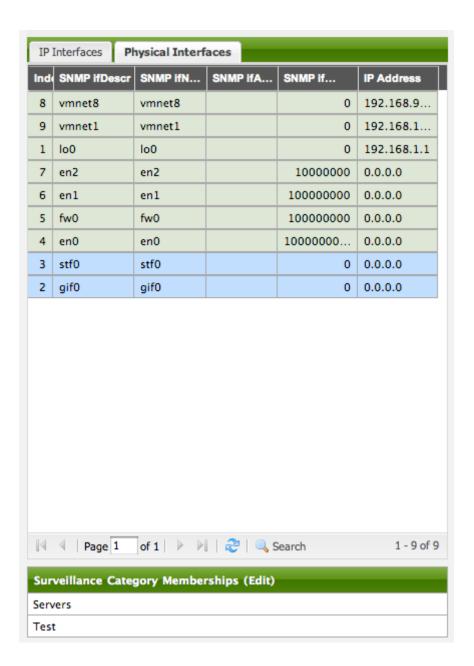




Following the import, All the IP and SNMP interfaces, in addition to the interface specified in the requisition, have been discovered and added to the node entity. The default foreign source definition has no polices for controlling which interfaces that are discovered either get persisted or managed by OpenNMS Meridian.



Logical and Physical interface and Service entities directed and discovered by Provisiond.



| Home / Search / Node / Interface Interface: 10.1.1.1 | | | | | | |
|--|--|--|--|--|--|--|
| /iew Events by Ip address View | Events by ifIndex Response Time Graphs Delete Rescan | | | | | |
| General | | | | | | |
| Node | p-brane | | | | | |
| Polling Status | Managed | | | | | |
| Polling Package | example1 | | | | | |
| Polling Package | strafer | | | | | |
| Interface Index | 1 | | | | | |
| Last Service Scan | 10/22/09 11:42:24 AM | | | | | |
| Physical Address | | | | | | |
| No link information has been collected for this interface. | | | | | | |
| Services | | | | | | |
| SNMP | | | | | | |
| SSH | | | | | | |
| ICMP | | | | | | |
| DNS | | | | | | |
| Availability | | | | | | |
| Overall Availability | 100.000% | | | | | |
| DNS | 100.000% | | | | | |
| ICMP | 100.000% | | | | | |
| SNMP | 100.000% | | | | | |
| SSH | 100.000% | | | | | |
| | Percentage over last 24 hour | | | | | |

Service Detection

As IP interfaces are found during the node scan process, service detection tasks are scheduled for each IP interface. The service detections defined in the foreign source determines which services are to be detected and how (i.e. the values of the parameters that parameters control how the service is detected, port, timeout, etc.).

Applying a New Foreign Source Definition

This example node has been provisioned using the Default foreign source definition. By navigating to the Provisioning Groups screen in the OpenNMS Meridian Web-UI and clicking the Edit Foreign Source link of a group, you can create a new foreign source definition that defines service detection and policies. The policies determine entity persistence and/or set attributes on the discovered entities that control OpenNMS Meridian management behaviors.

When creating a new foreign source definition, the default definition is used as a template.

: NMS

rval 1d

Add Detector

| me | DNS | class | org.opennms.netmgt.provision.detector.datagram.DnsDetector | [Add Paramete |
|----|--------|-------|---|---------------|
| me | FTP | class | org.opennms.netmgt.provision.detector.simple.FtpDetector | [Add Paramete |
| me | HTTP | class | org.opennms.netmgt.provision.detector.simple.HttpDetector | [Add Paramete |
| me | HTTPS | class | org.opennms.netmgt.provision.detector.simple.HttpsDetector | [Add Paramete |
| me | ICMP | class | org.opennms.netmgt.provision.detector.icmp.lcmpDetector | [Add Paramete |
| me | IMAP | class | org.opennms.netmgt.provision.detector.simple.lmapDetector | [Add Paramete |
| me | LDAP | class | org.opennms.netmgt.provision.detector.simple.LdapDetector | [Add Paramete |
| me | NRPE | class | org.opennms.netmgt.provision.detector.simple.NrpeDetector | [Add Paramete |
| me | POP3 | class | org.opennms.netmgt.provision.detector.simple.Pop3Detector | [Add Paramete |
| me | Radius | class | org.opennms.netmgt.provision.detector.radius.RadiusAuthDetector | [Add Paramete |
| me | SMTP | class | org.opennms.netmgt.provision.detector.simple.SmtpDetector | [Add Paramete |
| me | SNMP | class | org.opennms.netmgt.provision.detector.snmp.SnmpDetector | [Add Paramete |
| me | SSH | class | org.opennms.netmgt.provision.detector.ssh.SshDetector | [Add Paramete |
| | | | | |

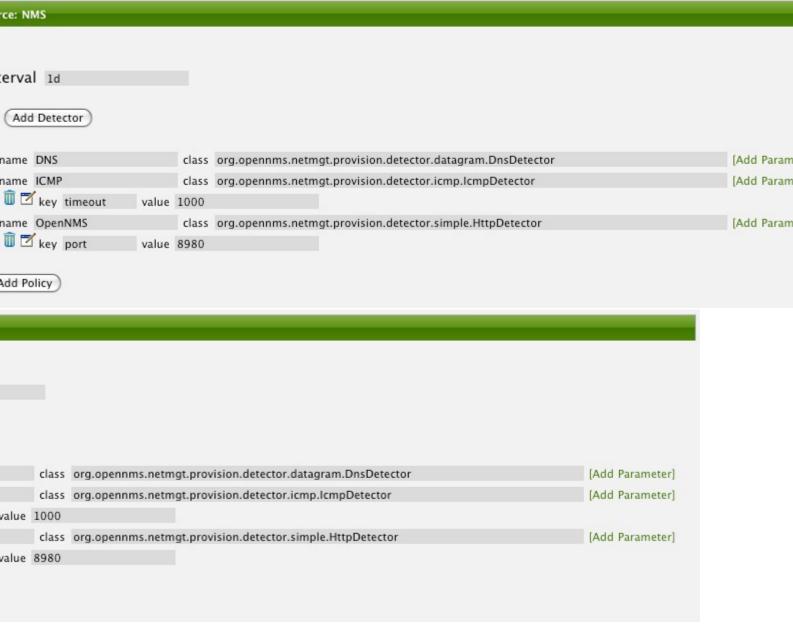
d Policy

| class | org.opennms.netmgt.provision.detector.datagram.DnsDetector | [Add Parameter] |
|-------|---|-----------------|
| class | org.opennms.netmgt.provision.detector.simple.FtpDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.simple.HttpDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.simple.HttpsDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.icmp.lcmpDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.simple.lmapDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.simple.LdapDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.simple.NrpeDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.simple.Pop3Detector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.radius.RadiusAuthDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.simple.SmtpDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.snmp.SnmpDetector | [Add Parameter] |
| class | org.opennms.netmgt.provision.detector.ssh.SshDetector | [Add Parameter] |
| | | |

In this UI, new Detectors can be added, changed, and removed. For this example, we will remove detection of all services accept ICMP and DNS, change the timeout of ICMP detection, and a new

Service detection for OpenNMS Meridian Web-UI.

Custom foreign source definition created for NMS Provisioning Group (foreign source).



Click the Done button and re-import the NMS Provisioning Group. During this and any subsequent re-imports or re- scans, the OpenNMS Meridian detector will be active, and the detectors that have been removed will no longer test for the related services for the interfaces on nodes managed in the provisioning group (requisition), however, the currently detected services will not be removed. There are 2 ways to delete the previously detected services:

- 1. Delete the node in the provisioning group, re-import, define it again, and finally re-import again
- 2. Use the ReST API to delete unwanted services. Use this command to remove each unwanted service from each interface, iteratively:

```
curl -X DELETE -H "Content-Type: application/xml" -u admin:admin
http://localhost:8980/opennms/rest/nodes/6/ipinterfaces/172.16.1.1/services/DNS
```



There is a sneaky way to do #1. Edit the provisioning group and just change the foreign ID. That will make Provisiond think that a node was deleted and a new node was added in the same requisition! Use this hint with caution and an full understanding of the impact of deleting an existing node.

Provisioning with Policies

The Policy API in Provisiond allow you to control the persistence of discovered IP and SNMP Interface entities and Node Categories during the Scan phase.

Matching IP Interface Policy

The Matching IP Interface policy controls whether discovered interfaces are to be persisted and if they are to be persisted, whether or not they will be forced to be Managed or Unmanaged.

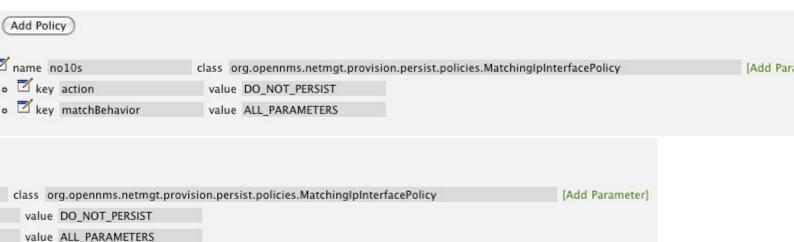
Continuing with this example Provisioning Group, we are going to define a few policies that:

- a. Prevent discovered 10 network addresses from being persisted
- b. Force 192.168 network addresses to be unmanaged

From the foreign source definition screen, click the Add Policy button and the definition of a new policy will begin with a field for naming the policy and a drop down list of the currently installed policies. Name the policy *no10s*, make sure that the *Match IP Interface policy* is specified in the class list and click the Save button. This action will automatically add all the parameters required for the policy.

The two required parameters for this policy are action and matchBehavior.

The action parameter can be set to DO_NOT_PERSIST, Manage, or UnManage.



Creating a policy to prevent persistence of 10 network IP interfaces.

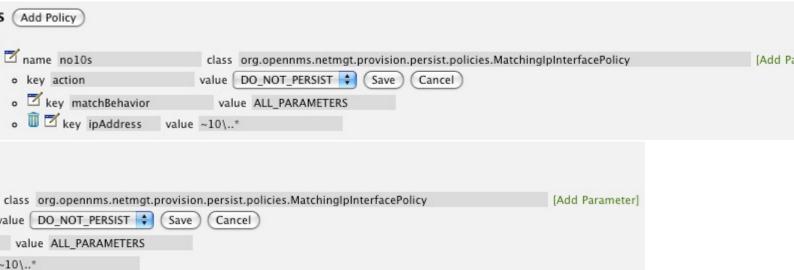
The *DO_NOT_PERSIST* action does just what it indicates, it prevents discovered IP interface entities from being added to OpenNMS Meridian when the *matchBehavior* is satisfied. The Manage and UnManage values for this action allow the IP interface entity to be persisted by control whether or not that interface should be managed by OpenNMS Meridian.

The matchBehavior action is a boolean control that determines how the optional parameters will

be evaluated. Setting this parameter's value to *ALL_PARAMETERS* causes *Provisiond* to evaluate each optional parameter with boolean *AND* logic and the value *ANY_PARAMETERS* will cause *OR* logic to be applied.

Now we will add one of the optional parameters to filter the 10 network addresses. The Matching IP Interface policy supports two additional parameters, *hostName* and *ipAddress*. Click the *Add Parameter* link and choose *ipAddress* as the *key*. The *value* for either of the optional parameters can be an exact or regular expression match. As in most configurations in OpenNMS Meridian where regular expression matching can be optionally applied, prefix the value with the ~ character.

Example Matching IP Interface Policy to not Persist 10 Network addresses

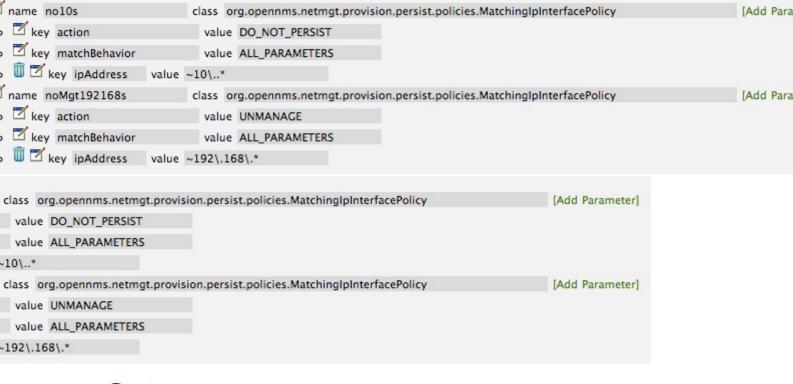


Any subsequent scan of the node or re-imports of NMS provisioning group will force this policy to be applied. IP Interface entities that already exist that match this policy will not be deleted. Existing interfaces can be deleted by recreating the node in the *Provisioning Groups* screen (simply change the foreign ID and re-import the group) or by using the ReST API:

```
curl -X DELETE -H "Content-Type: application/xml" -u admin:admin
http://localhost:8980/opennms/rest/nodes/6/ipinterfaces/10.1.1.1
```

The next step in this example is to define a policy that sets discovered 192.168 network addresses to be unmanaged (not managed) in OpenNMS Meridian. Again, click the Add Policy button and let's call this policy *noMgt192168s*. Again, choose the Mach IP Interface policy and this time set the action to *UNMANAGE*.

Policy to not manage IP interfaces from 192.168 networks



0

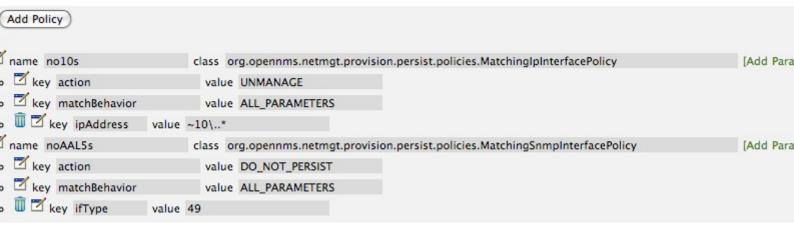
The *UNMANAGE* behavior will be applied to existing interfaces.

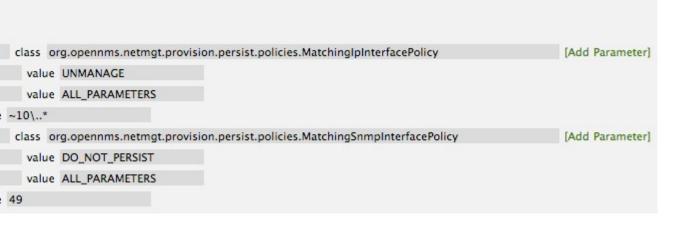
Matching SNMP Interface Policy

Like the Matching IP Interface Policy, this policy controls the whether discovered SNMP interface entities are to be persisted and whether or not OpenNMS Meridian should collect performance metrics from the SNMP agent for Interface's index (MIB2 IfIndex).

In this example, we are going to create a policy that doesn't persist interfaces that are *AAL5* over *ATM* or type 49 (*ifType*). Following the same steps as when creating an IP Management Policy, edit the foreign source definition and create a new policy. Let's call it: *noAAL5s*. We'll use Match SNMP Interface class for each policy and add a parameter with *ifType* as the key and 49 as the value.

Matching SNMP Interface Policy example for Persistence and Data Collection





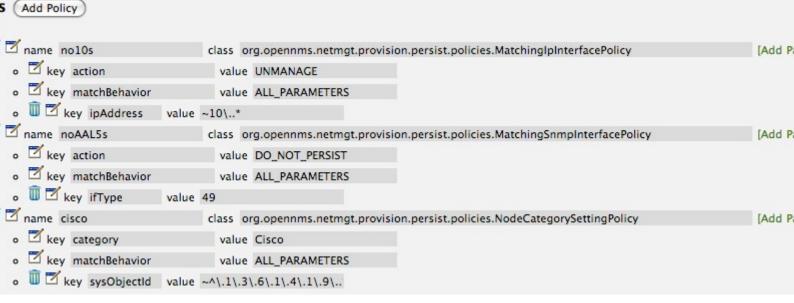


At the appropriate time during the scanning phase, Provisiond will evaluate the policies in the foreign source definition and take appropriate action. If during the policy evaluation process any policy matches for a "DO_NOT_PERSIST" action, no further policy evaluations will happen for that particular entity (IP Interface, SNMP Interface).

Node Categorization Policy

With this policy, nodes entities will automatically be assigned categories. The policy is defined in the same manner as the IP and SNMP interface polices. Click the Add Policy button and give the policy name, cisco and choose the *Set Node Category* class. Edit the required *category* key and set the value to Cisco. Add a policy parameter and choose the *sysObjectId* key with a value $^{\wedge}$ 1\.3\.6\.1\.4\.1\.9\..*.

Example: Node Category setting policy



| class org.opennms.netmgt.provision.persist.policies.MatchinglpInterfacePolicy | [Add Parameter] |
|---|-----------------|
| value UNMANAGE | |
| value ALL_PARAMETERS | |
| ~10* | |
| class org.opennms.netmgt.provision.persist.policies.MatchingSnmpInterfacePolicy | [Add Parameter] |
| value DO_NOT_PERSIST | |
| value ALL_PARAMETERS | |
| 49 | |
| class org.opennms.netmgt.provision.persist.policies.NodeCategorySettingPolicy | [Add Parameter] |
| value Cisco | |
| value ALL_PARAMETERS | |
| ~^\.1\.3\.6\.1\.4\.1\.9\ | |

New Import Capabilities

Several new XML entities have been added to the import requisition since the introduction of the OpenNMS Importer service in version 1.6. So, in addition to provisioning the basic node, interface, service, and node categories, you can now also provision asset data.

Provisiond Configuration

The configuration of the Provisioning system has moved from a properties file (model-importer.properties) to an XML based configuration container. The configuration is now extensible to allow the definition of 0 or more import requisitions each with their own *Cron* based schedule for automatic importing from various sources (intended for integration with external URL such as HTTP and this new DNS protocol handler.

A default configuration is provided in the OpenNMS Meridian etc/ directory and is called: provisiond-configuration.xml. This default configuration has an example for scheduling an import from a DNS server running on the localhost requesting nodes from the zone, localhost and will be imported once per day at the stroke of midnight. Not very practical but is a good example.

```
<?xml version="1.0" encoding="UTF-8"?>
   xsi:schemaLocation="http://xmlns.opennms.org/xsd/config/provisiond-configuration"
       foreign-source-dir="/opt/opennms/etc/foreign-sources"
       requistion-dir="/opt/opennms/etc/imports"
       importThreads="8"
       scanThreads="10"
       rescanThreads="10"
       writeThreads="8" >
   <!--
       http://www.quartz-scheduler.org/documentation/quartz-
1.x/tutorials/crontrigger[http://www.quartz-scheduler.org/documentation/quartz-
1.x/tutorials/crontrigger]
       Field Name Allowed Values Allowed Special Characters
       Seconds 0-59 , - * / Minutes 0-59 , - * / Hours 0-23 , - * /
       Day-of-month1-31, - * ? / L W C Month1-12 or JAN-DEC, - * /
       Day-of-Week1-7 or SUN-SAT, - * ? / L C # Year (Opt)empty, 1970-2099, - * /
   -->
   <requisition-def import-name="NMS"
                   import-url-resource="file://opt/opennms/etc/imports/NMS.xml">
       <cron-schedule>0 0 0 * * ? *</cron-schedule> <!-- daily, at midnight -->
   </requisition-def>
</provisiond-configuration>
```

Configuration Reload

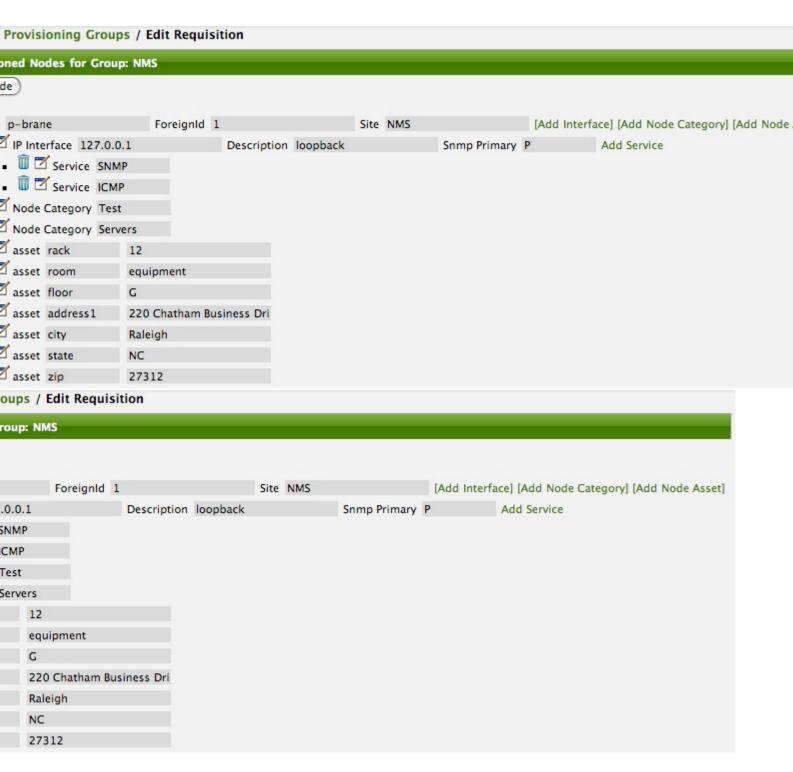
Like many of the daemon configurations in the 1.7 branch, *Provisiond's* configuration is re-loadable without having to restart OpenNMS. Use the reloadDaemonConfig uei:

```
/opt/opennms/bin/send-event.pl uei.opennms.org/internal/reloadDaemonConfig --parm
'daemonName Provisiond'
```

This means that you don't have to restart OpenNMS Meridian every time you update the configuration!

Provisioning Asset Data

The Provisioning Groups Web-UI had been updated to expose the ability to add Node Asset data in an import requisition. Click the *Add Node Asset* link and you can select from a drop down list all the possible node asset attributes that can be defined.



After an import, you can navigate to the *Node Page* and click the *Asset Info* link and see the asset data that was just provided in the requisition.

| | | | Division | | | | Department | |
|----------|---------------------------|----------|----------|----|------------|-----------|------------|-----------|
| 220 (| Chatham Business Drive, S | uite 100 |) | | | | | |
| | | | | | | | | |
| Raleig | gh | | State | NC | | | ZIP | 27312 |
| NMS | | | Floor | G | | | Room | equipment |
| 12 | | | Slot | | | | Port | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | Division | | | | Department | | | |
| Suite 10 | 0 | | | | | | | |
| | | | | | | | | |
| | State | NC | | | ZIP | 27312 | | |
| | Floor | G | | | Room | equipment | | |
| | Slot | | | | Port | | | |
| | | | | | | | | |

External Requisition Sources

Because Provisiond takes a *URL* as the location service for import requisitions, OpenNMS Meridian can be easily extended to support sources in addition to the native URL handling provided by Java: *file://, http://,* and *https://.* When you configure *Provisiond* to import requisitions on a schedule you specify using a *URL* Resource. For requisitions created by the *Provisioning Groups* WebUI, you can specify a file based URL.



<need further documentation>

Provisioning Nodes from DNS

The new Provisioning service in OpenNMS Meridian is continuously improving and adapting to the needs of the community. One of the most recent enhancements to the system is built upon the very flexible and extensible API of referencing an import requisition's location via a URL. Most commmonly, these URLs are files on the file system (i.e. file:/opt/opennms/etc/imports/<my-provisioning-group.xml>) as requisitions created by the Provisioning Groups UI. However, these same requisitions for adding, updating, and deleting nodes (based on the original model importer) can also come from URLs specifying the HTTP protocol: http://myinventory.server.org/nodes.cgi)

Now, using Java's extensible protocol handling specification, a new protocol handler was created so

that a URL can be specified for requesting a Zone Transfer (*AXFR*) request from a DNS server. The *A records* are recorded and used to build an import requisition. This is handy for organizations that use DNS (possibly coupled with an IP management tool) as the data base of record for nodes in the network. So, rather than ping sweeping the network or entering the nodes manually into OpenNMS Meridian Provisioning UI, nodes can be managed via 1 or more DNS servers. The format of the URL for this new protocol handler is:

```
dns://<host>[:port]/<zone>[/<foreign-source>/][?expression=<regex>]
```

Simple Example

```
dns://my-dns-server/myzone.com
```

This will import all *A records* from the host *my-dns-server* on port 53 (default port) from zone *myzone.com* and since the foreign source (a.k.a. the provisioning group) is not specified it will default to the specified zone.

Using a Regular Expression Filter

You can also specify a subset of the *A records* from the zone transfer using a regular expression:

```
dns://my-dns-server/myzone.com/portland/?expression=^por-.*
```

This will import all nodes from the same server and zone but will only manage the nodes in the zone matching the regular expression 'port-.* and will and they will be assigned a unique foreign source (provisioning group) for managing these nodes as a subset of nodes from within the specified zone.

URL Encoding

If your expression requires URL encoding (for example you need to use a ? in the expression) it must be properly encoded.

```
dns://my-dns-server/myzone.com/portland/?expression=^por[0-9]%3F
```

DNS Setup

Currently, the DNS server requires to be setup to allow a zone transfer from the OpenNMS Meridian server. It is recommended that a secondary DNS server is running on OpenNMS Meridian and that the OpenNMS Meridian server be allowed to request a zone transfer. A quick way to test if zone transfers are working is:

```
dig -t AXFR @<dn5Server> <zone>
```

9.6. Adapters

The OpenNMS Meridian *Provisiond API* also supports *Provisioning Adapters* (plugins) for integration with external systems during the provisioning Import phase. When node entities are added, updated, deleted, or receive a configuration management change event, OpenNMS Meridian will call the adapter for the provisioning activities with integrated systems.

Currently, OpenNMS Meridian supports the following adapters:

9.6.1. DDNS Adapter

The Opposite end of *Provisiond* integration from the DNS Requisition Import, is the *DDNS adapter*. This adapter uses the *dynamic DNS protocol* to update a DNS system as nodes are provisioned into OpenNMS Meridian. To configure this adapter, edit the opennms.properties file and set the importer.adapter.dns.server property:

importer.adapter.dns.server=192.168.1.1

9.6.2. RANCID Adapter

Integration has been integrated with RANCID though this new API.



<More documentation needed>



Maps (soon to be moved to Mapd) <documentation required>



WiMax-Link (soon to be moved to Linkd) <documentation required>

9.7. Integrating with Provisiond

The ReST API should be used for integration from other provisioning systems with OpenNMS Meridian. The ReST API provides an interface for defining foreign sources and requisitions.

9.7.1. Provisioning Groups of Nodes

Just as with the WebUI, groups of nodes can be managed via the ReST API from an external system. The steps are:

- 1. Create a Foreign Source (if not using the default) for the group
- 2. Update the SNMP configuration for each node in the group
- 3. Create/Update the group of nodes

9.7.2. Example

Step 1 - Create a Foreign Source

If policies for this group of nodes are going to be specified differently than the default policy, then a foreign source should be created for the group. Using the ReST API, a foreign source can be provided. Here is an example:



The XML can be imbedded in the curl command option -d or be referenced from a file if the @ prefix is used with the file name as in this case.

The XML file: customer-a.foreign-source.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<foreign-source date-stamp="2009-10-12T17:26:11.616-04:00" name="customer-a" xmlns=</pre>
"http://xmlns.opennms.org/xsd/config/foreign-source">
    <scan-interval>1d</scan-interval>
    <detectors>
        <detector class="orq.opennms.netmqt.provision.detector.icmp.IcmpDetector"</pre>
name="ICMP"/>
        <detector class="org.opennms.netmgt.provision.detector.snmp.SnmpDetector"</pre>
name="SNMP"/>
   </detectors>
    <policies>
        <policy class=</pre>
"org.opennms.netmgt.provision.persist.policies.MatchingIpInterfacePolicy" name="no-
192-168">
            <parameter value="UNMANAGE" key="action"/>
            <parameter value="ALL_PARAMETERS" key="matchBehavior"/>
            <parameter value="~^192\.168\..*" key="ipAddress"/>
        </policy>
    </policies>
</foreign-source>
```

Here is an example curl command used to create the foreign source with the above foreign source specification above:

```
curl -v -u admin:admin -X POST -H 'Content-type: application/xml' -d '@customer-a.foreign-source.xml' http://localhost:8980/opennms/rest/foreignSources
```

Now that you've created the foreign source, it needs to be deployed by Provisiond. Here an the example using the curl command to deploy the foreign source:

```
curl -v -u admin:admin
http://localhost:8980/opennms/rest/foreignSources/pending/customer-a/deploy -X PUT
```



The current API doesn't strictly follow the ReST design guidelines and will be updated in a later release.

Step 2 - Update the SNMP configuration

The implementation only supports a *PUT* request because it is an implied "Update" of the configuration since it requires an IP address and all IPs have a default configuration. This request is is passed to the SNMP configuration factory in OpenNMS Meridian for optimization of the configuration store snmp-config.xml. This example changes the community string for the IP address 10.1.1.1 to yRuSonoZ.



Community string is the only required element

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

Step 3 - Create/Update the Requisition

This example adds 2 nodes to the Provisioning Group, *customer-a*. Note that the foreign-source attribute typically has a 1 to 1 relationship to the name of the Provisioning Group requisition. There is a direct relationship between the foreign-source attribute in the requisition and the foreign source policy specification. Also, typically, the name of the provisioning group will also be the same. In the following example, the ReST API will automatically create a provisioning group based on the value foreign-source attribute specified in the XML requisition.

```
curl -X POST -H "Content-Type: application/xml" -d "<?xml version="1.0" encoding="UTF-8"?><model-import xmlns="http://xmlns.opennms.org/xsd/config/model-import" date-stamp="2009-03-07T17:56:53.123-05:00" last-import="2009-03-07T17:56:53.117-05:00" foreign-source="customer-a"><node node-label="p-brane" foreign-id="1" ><interface ip-addr="10.0.1.3" descr="en1" status="1" snmp-primary="P"><monitored-service service-name="ICMP"/><monitored-service service-name="SNMP"/></interface><category name="Production"/><category name="Routers"/></node><node node-label="m-brane" foreign-id="1" ><interface ip-addr="10.0.1.4" descr="en1" status="1" snmp-primary="P"><monitored-service service-name="ICMP"/><monitored-service service-name="ICMP"/></interface><category name="Production"/><category name="Production"/><category name="Routers"/></node></model-import>" -u admin:admin http://localhost:8980/opennms/rest/requisitions
```

A provisioning group file called etc/imports/customer-a.xml will be found on the OpenNMS Meridian system following the successful completion of this curl command and will also be visible via the WebUI.



Add, Update, Delete operations are handled via the ReST API in the same manner as described in detailed specification.

9.8. Provisioning Single Nodes (Quick Add Node)

Adding a Node to a Current Requisition

Often, it is requested that a single node add/update be completed for an already defined provisioning group. There is a ReST API for the *Add Node* implementation found in the OpenNMS Meridian Web-UI. For this to work, the provisioning group must already exist in the system even if there are no nodes defined in the group.

- 1. Create a foreign source (if required)
- 2. Specify SNMP configuration
- 3. Provide a single node with the following specification

9.9. Fine Grained Provisioning Using provision.pl

provision.pl provides an example command-line interface to the provisioning-related OpenNMS Meridian REST API endpoints.

The script has many options but the first 3 optional parameters are described here:



You can use --help to the script to see all the available options.

```
--username (default: admin)
--password (default: admin)
--url (default: http://localhost:8980/opennms/rest)
```

9.9.1. Create a new requisition

provision.pl provides easy access to the requisition REST service using the *requisition* option:

```
${OPENNMS_HOME}/bin/provision.pl requisition customer1
```

This command will create a new, empty (containing no nodes) requisition in OpenNMS Meridian.

The new requisition starts life in the pending state. This allows you to iteratively build the requisition and then later actually import the nodes in the requisition into OpenNMS Meridian. This handles all adds/changes/deletes at once. So, you could be making changes all day and then at night either have a schedule in OpenNMS Meridian that imports the group automatically or you can send a command through the REST service from an outside system to have the pending requisition imported/reimported.

You can get a list of all existing requisitions with the list option of the provision.pl script:

```
${OPENNMS_HOME}/bin/provision.pl list
```

Create a new Node

```
${OPENNMS_HOME}/bin/provision.pl node add customer1 1 node-a
```

This command creates a node element in the requisition *customer1* called *node-a* using the script's *node* option. The node's foreign-ID is 1 but it can be any alphanumeric value as long as it is unique within the requisition. Note the node has no interfaces or services yet.

Add an Interface Element to that Node

```
${OPENNMS_HOME}/bin/provision.pl interface add customer1 1 127.0.0.1
```

This command adds an interface element to the node element using the *interface* option to the provision.pl command and it can now be seen in the pending requisition by running provision.pl requisition list customer1.

Add a Couple of Services to that Interface

```
${OPENNMS_HOME}/bin/provision.pl service add customer1 1 127.0.0.1 ICMP ${OPENNMS_HOME}/bin/provision.pl service add customer1 1 127.0.0.1 SNMP
```

This adds the 2 services to the specified 127.0.0.1 interface and is now in the pending requisition.

Set the Primary SNMP Interface

```
${OPENNMS_HOME}/bin/provision.pl interface set customer1 1 127.0.0.1 snmp-primary P
```

This sets the 127.0.0.1 interface to be the node's Primary SNMP interface.

Add a couple of Node Categories

```
${OPENNMS_HOME}/bin/provision.pl category add customer1 1 Routers
${OPENNMS_HOME}/bin/provision.pl category add customer1 1 Production
```

This adds the two categories to the node and is now in the pending requisition.

These categories are case-sensitive but do not have to be already defined in OpenNMS Meridian. They will be created on the fly during the import if they do not already exist.

Setting Asset Fields on a Node

```
${OPENNMS_HOME}/bin/provision.pl asset add customer1 1 serialnumber 9999
```

This will add value of 9999 to the asset field: serialnumber.

Deploy the Import Requisition (Creating the Group)

```
${OPENNMS_HOME}/bin/provision.pl requisition import customer1
```

This will cause OpenNMS Meridian Provisiond to import the pending customer1 requisition. The formerly pending requisition will move into the deployed state inside OpenNMS Meridian.

Deleting a Node from a Requisition

Very much the same as the add, except that a single delete command and a re-import is required. What happens is that the audit phase is run by Provisiond and it will be determined that a node has been removed from the requisition and the node will be deleted from the DB and all services will stop activities related to it.

```
${OPENNMS_HOME}/bin/provision.pl node delete customer1 1 node-a ${OPENNMS_HOME}/bin/provision.pl requisition import customer1
```

This completes the life cycle of managing a node element, iteratively, in a import requisition.

9.10. Yet Other API Examples

List the Nodes in a Provisioning Group

The provision.pl script doesn't supply this feature but you can get it via the REST API. Here is an example using curl:

```
#!/bin/bash
REQ=$1
curl -X GET -H "Content-Type: application/xml" -u admin:admin
http://localhost:8980/opennms/rest/requisitions/$REQ 2>/dev/null | xmllint --format -
```

9.11. Service Detectors

9.11.1. SNMP Detector

This detector is used to find and assigns services based on *SNMP*. The detector binds a service with a given *Service Name* when a particular *SNMP OID* as scalar or table matches a given criteria.

Detector facts

```
Implementation org.opennms.netmgt.provision.detector.snmp.SnmpDetector
```

Configuration and Usage

Table 79. Parameters for the SNMP detector

| Param eter | Description | Requi red | Default value |
|---------------|---|--------------|---------------------|
| oid | SNMP OID for scalar or table to detect the service. | requir ed | .1.3.6.1.2.1 .1.2.0 |
| retry | Number of retries to detect the service. | option al | agent config |
| timeout | Timeout in milliseconds to wait for a response from the <i>SNMP</i> agent. | option al | agent config |
| vbvalue | expected return value to detect the service; if not specified the service is detected if the <i>SNMP OID</i> returned any kind of valid value. The vbvalue is evaluated as Java Regular Expression. | option al | - |
| hex | Set true if the data is from type HEX-String. | option al | false |
| isTable | Set true if detector should evaluate SNMP tables. | option al | false |
| matchTy pe | Set match type to evaluate the expected value in the SNMP table. <i>EXIST</i> : the expected vbalue is ignored, service detected if the given table under <i>OID</i> exist <i>ALL</i> : all values in the table must match against expected vbalue to detect service <i>ANY</i> : at least one value in the table must match against expected vbalue to detect service <i>NONE</i> : None of the values should match against expected value to detect service | option al | EXIST |

Example for SNMP scalar value

We have *Dell* server farm and want to monitor the global server status provided by the *OpenManage Server Administrator*. Global status is provided by a scalar *OID* .1.3.6.1.4.1.674.10892.1.200.10.1.2.1. The service should be automatically detected if the server supports this *OID*.

For provisioning we have a requisition named *Server* which contains all server of our data center. A *Detector* with the name *Dell-OMSA-Global-State* for this requisition is created with the following parameter:

Table 80. Parameters for the SNMP detector

| Paramete r | Value |
|---------------|---------------------------------------|
| Name | Dell-OMSA-Global-State |
| oid | .1.3.6.1.4.1.674.10892.1.200.10.1.2.1 |

When the requisition *Server* is synchronized the service *Dell-OMSA-Global-State* will be detected in case they support the given *SNMP OID*.

Example using SNMP tables

We have a *HP* server farm and want to monitor the status of logical drives over *SNMP* provided from *HP Insight Manager*. The status for logical drives is provided in a *SNMP Table* under .1.3.6.1.4.1.232.3.2.3.1.1.4. The service should be automatically assigned to all servers exposing the given *SNMP OID*.

For provisioning we have a requisition named *Server* which contains all server of our data center. A *Detector* with the name *HP-Insight-Drive-Logical* for this requisition is created with the following parameter:

Table 81. Parameters for the SNMP detector

| Paramete r | Value |
|---------------|------------------------------|
| Name | HP-Insight-Drive-Logical |
| oid | .1.3.6.1.4.1.232.3.2.3.1.1.4 |
| isTable | true |

When the requisition *Server* is synchronized the service *HP-Insight-Drive-Logical* will be detected in case they support the given *SNMP OID* table.

Chapter 10. Business Service Monitoring

This section describes how to model and configure *Business Services (BS)* and orchestrate them in a hierarchy. The concepts and usage of the section *Business Service Monitoring* from the *User Guide* is presumed.

Business Service Monitoring (BSM) includes the following components:

- Business Service Monitoring Daemon (BSMD): Maintains and drives the state of all BS
- Business Service Editor: Web application which allows you to create, update or delete BS
- Topology View for Business Services: Visual representation of the Business Service Hierarchy as a component of the Topology User Interface.
- BSM ReST API: ReST based API to create, read, update or delete BS

10.1. Business Service Definition

The status of *Service Monitors* and any kind of *Alarm* can be used to drive the *Operational Status* of a *BS*. A *BS* is defined with the following components:

- Business Service Name: A unique name used to identify the BS
- Edges: A set of elements on which this BS relies which can include other BS, or Reduction Keys.
- *Reduce Function*: Function used to aggregate the *Operational Status* from all the *Edges*. Specific functions may take additional parameters.
- *Attributes*: Optional key/value pairs that can be used to tag or enrich the *Busines Service* with additional information.

Each *Business Service* can contain a list of optional key/value attributes. These can be used to identify or tag the *BS*, and may be reference in other workflows. These attributes do not affect the dependencies or the status calculation of the *BS*.



Attributes can be used to filter BS in Ops Board dashlets.

The Business Service Editor is used to manage and model the Business Services and their hierarchy. It is required to have administrative permissions and is available in "Login Name \rightarrow Configure OpenNMS \rightarrow Manage Business Services" in the Service Monitoring section.

Managing Business Services with the Business Service Editor

- ① Create a new Business Service definition
- ② Collapse tree view for all Business Services in the view
- 3 Expand tree view for all Business Services in the view
- 4 Reload all Business Services in the view with current Business Services from the system
- ⑤ Reload the Business Service Monitoring Daemon to use the Business Service definition as configured
- 6 Business Service dependency hierarchy as tree view
- ① Show the current Business Service with dependencies in the Topology UI
- 8 Edit and delete existing Business Service defintions

As shown in figure Managing Business Services with the Business Service Editor the Business Services can be created or changed. The hierarchy is created by assigning an existing Business Service as Child Service.

10.2. Edges

Edges map the Alarm status monitoring with OpenNMS

The following types can be used:

- Child Service: A reference to an existing Business Service on which to depend
- *IP Service*: A convenient way to refer to the alarms that can be generated by a monitored *IP Service*. This will automatically provided edges for the *nodeLostService*, *interfaceDown* and *nodeDown* reductions keys of the specified service.
- Reduction Key: A resolved Reduction Key used to refer to a specific Alarm, e.g. generated by a SNMP Trap or Threshold violation



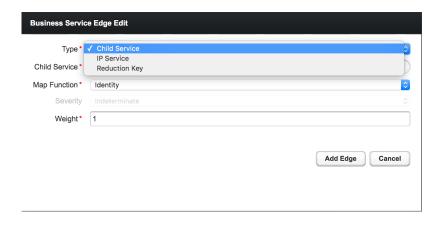
If you need help determining the reduction key used by alarm, trigger the alarm in question and pull the reduction key from the *Alarm* details page.

All edge types have the following parameters:

- Map Function: The associated Map Function for this Edge
- Weight: The relative Weight of this edge. Used by certain Reduce Functions.

Both *IP Service* and *Reduction Key* type edges also support a *Friendly Name* parameter which gives the user control on how the edge is labeled in the *Topology User Interface*. The editor changing the *Edge* attributes is shown in figure Editor to add Business Service Edges.

Editor to add Business Service Edges



10.2.1. Child Services

To create a hierarchy of *Business Services* they need to be created first. The hierarchy is build by selecting the *Business Service* as Child Service_ as dependency.

10.2.2. IP Services

The *IP Service* is a predefined set of *Reduction Keys* which allows easily to assign a specific *Monitored Service* to the given *BS*. As an example you have multiple Servers with a *Monitored Service SMTP* and you want to model a *BS* named *Mail Communication*. If just the *Reduction Key* for a *nodeLostService* is assgined, the *BS* would not be affected in case the *IP Interface* or the whole *Node* goes down. *OpenNMS* generates *Alarms* with different *UEI* which needs to be assigned to the *BS* as well. To make it easier to model this use case the *IP Service* generates the following *Reduction Keys* automatically:

- uei.opennms.org/nodes/nodeLostService:%nodeId%:%ipAddress%:%serviceName%: Matches *Alarms* when the given *Monitored Service* goes down
- uei.opennms.org/nodes/interfaceDown:%nodeId%:%ipAddress%: Matches *Alarms* when the given *IP Interface* of the *Monitored Service* goes down
- uei.opennms.org/nodes/nodeDown:%nodeId%: Matches *Alarms* when the given *Node* of the *Monitored Service* goes down

10.2.3. Custom Reduction Key

The *Reduction Key* edge is used to refer to specific instance of alarms. When an alarm with the given *Reduction Key* is present, the alarms' severity will be used to calculate the *Operational Status* of the *BS*. To give a better explanation a *Friendly Name* can be set and is used in the *Business Service View*. The format of the *Reduction Key* is build by a set of attributes as a key separated by : and enclosed in %, i.e (%attribute*:%attribute*).

Example of a Reduction Key for a specific nodeLostService

%uei.opennms.org/nodes/nodeLostService%:%nodeId%:%ipAddress%:%serviceName%

10.3. Map Functions

The *Map Functions* define how the *Severity* of the edge will be used in the *Reduce Function* of the parent when calculating the *Operational Status*.

The available *Map Functions* are:

Table 82. Calculation of the Operational Status with Map Functions

| Name | Description |
|----------|--|
| Identity | Use the same Severity as Operational Status of the BS |
| Increase | Increase the Severity by one level and use it as Operational Status of the BS |
| Decrease | Decrease the <i>Severity</i> by one level and use it as <i>Operational Status</i> of the <i>BS</i> |
| SetTo | Set the <i>Operational Status</i> to a constant <i>Severity</i> value |
| Ignore | The input of the <i>Edge</i> is ignored for <i>Operational Status</i> calculation |

10.4. Reduce Functions

A *Reduce Function* is used to aggregate the *Operational Status* for the *BS.* The *Alarm Severity* from the *Edges* are used as input for the *Reduce Function*. For this operation the following *Reduce Functions* are available:

Table 83. Status calculation Reduce Functions

| Name | Description |
|---------------------------|---|
| Highest Severity | Uses the value of the highest severity, Weight is ignored. |
| Threshold | Uses the highest severity found more often than the given threshold, e.g. 0.26 can also be seen as 26%, which means at least 2 of 4 <i>Alarms</i> need to be raised to change the <i>BS</i> . |
| Highest Severity Above | Uses the highest severity greater than the given threshold severity. |

| Name | Description |
|-------------------------|--|
| Exponential Propagation | This reduce function computes the sum of the given child severities based on a base number. For this computation the severities are mapped to numbers: WARNING=0, MINOR=1, MAJOR=2, CRITICAL=3 All other severities are ignored. For the aggregation the following formula will be used to compute the resulting Business Service severity from its n child entities based on the base number b : severity = $ _\log_{b}(sum_{i=1})^n b^(ch idSeverity_{i}) $ In summary the base value defines how many items of a severity x will result in a severity $x+1$. Results lower as a are treated as a a are treated as a a are treated as a a a are treated as a |

The following table shows the status calculation with *Edges* assigned to an *IP Service*. The *IP-Service* is driven by the monitoring of the *ICMP* service for three Web Server. In the table below you find a configuration where *Web Server 3* is weighted 3 times higher than the other and a threshold of 0.33 (33%) is configured.

Table 84. Example for status calculation using the Threshold function

| Name | Weig ht | Weight Factor | Input Severity | Operational Status | Critic al | Majo r | Mino r | Warni ng | Norm al |
|----------------|------------|------------------|-------------------|-----------------------|--------------|-----------|-----------|-------------|------------|
| Web- ICMP-1 | 1 | 0.2 | Critical | Critical | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| Web- ICMP-2 | 1 | 0.2 | Normal | Normal | 0 | 0 | 0 | 0 | 0.2 |
| Web- ICMP-3 | 3 | 0.6 | Warning | Warning | 0 | 0 | 0 | 0.6 | 0.6 |
| Total | | 1.0 | | | 0.2 | 0.2 | 0.2 | 0.8 | 1 |
| Percentag e | | 100% | | | 20% | 20% | 20% | 80% | 100% |

The *Operational Status Severity* is evaluated from left to right, the first value higher then the configured *Threshold* is used. In this case the *Operational Status* is set to *Warning* because the first threshold which exceeds 33% is *Warning* with 80%.

10.5. Business Service Daemon

The calculation of the Operational Status of the BS is driven by the Business Service Monitoring

Daemon (bsmd). The daemon is responsible for tracking the operational status of all *BS* and for sending events in case of operational status changes.

In order to calculate the *Operational Status* the *reduction key* associated with a *Business Service* is used. The *reduction key* is obtained from an alarm generated by *OpenNMS Meridian*. This means that the alarm's *reduction key* of a defined *Business Service* must not change afterwards. Otherwise *bsmd* is not able to calculate the *Operational Status* correctly. This also applies for removing the *alarm data* from events associated to *Business Services* In addition the child type "IP Service" from the *Business Service Configuration Page* requires the following events with the default reduction keys being defined:

* uei.opennms.org/nodes/nodeLostService * uei.opennms.org/nodes/nodeSpondeNodeDown * uei.opennms.org/nodes/interfaceDown

Every time the configuration of a *Business Service* is changed a reload of the daemon's configuration is required. This includes changes like the name of the *Business Service* or its attributes as well as changes regarding the *Reduction Keys*, contained *Business Services* or *IP Services*. The *bsmd* configuration can be reloaded with the following mechanisms:

- Click the Reload Daemon button in the Business Service Editor
- Send the *reloadDaemonConfig* event using send-event.pl or use the WebUI in *Manually Send an Event* with parameter daemonName bsmd
- Use the ReST API to perform a POST request to /opennms/api/v2/business-services/daemon/reload

If the reload of the configuration is done an event of type uei.opennms.org/internal/reloadDaemonConfigSuccessful is fired.

Example reloading bsmd configuration from CLI

\$OPENNMS_HOME/bin/send-event.pl -p 'daemonName bsmd'
uei.opennms.org/internal/reloadDaemonConfig

Example reloading bsmd configuration through ReST POST

curl -X POST -u admin:admin -v http://localhost:8980/opennms/api/v2/business-services/daemon/reload

Chapter 11. Topology Map

This section describes how to configure the *Topology Map*.

11.1. Properties

The *Topology Map* supports the following properties, which can be influenced by changing the file etc/org.opennms.features.topology.app.cfg:

| Property | Туре | Default | Description |
|--------------------------|-------------|-----------------|---|
| showHeader | Boole an | true | Defines if the <i>OpenNMS Meridian</i> header is shown. |
| autoRefresh.e nabled | Boole an | false | If enabled, auto refresh is enabled by default. |
| autoRefresh.i nterval | Integ er | 60 | Defines the auto refresh interval in seconds. |
| hiddenCategor yPrefix | Strin g | empty String | A String which allows hiding categories. For example a value of server will hide all categories starting with server. Be aware, that this setting is case-sensistive, so Servers will be shown. The resolution is only enabled if no longitude/latitude information is available. |

11.2. Icons

Each Vertex on the *Topology Map* is represented by an icon. The default icon is configured in the icon mapping file: \${OPENNMS_HOME}/etc/org.opennms.features.topology.app.icons.<topology-namespace>.cfg. If an icon mapping file does not exist for a *Topology Provider*, the provider does not support customization.

List of available icon mapping files (may not be complete)

```
org.opennms.features.topology.app.icons.default.cfg ①
org.opennms.features.topology.app.icons.application.cfg ②
org.opennms.features.topology.app.icons.bsm.cfg ③
org.opennms.features.topology.app.icons.linkd.cfg ④
org.opennms.features.topology.app.icons.vmware.cfg ⑤
```

- 1 Default icon mapping
- 2 Icon mapping for the Application Topology Provider
- 3 Icon mapping for the Business Services Topology Provider
- 4 Icon mapping for the Linkd Topology Provider
- 5 Icon mapping for the Vmware Topology Provider

Each File contains a mapping in form of <icon key> = <icon id>.

Icon key

A *Topology Provider* dependent string which maps to an icon id. An icon key consists of one to multiple segments. Each segment must contain only numbers or characters. If multiple segments exist they must be separated by ., e.g. my.custom.key. Any existing default icon keys are not configurable and should not be changed.

Icon id

The icon id is a unique icon identifier to reference an icon within one of the available SVG icons located in \${OPENNMS_HOME}/jetty-webapps/opennms/svg. For more details see Add new icons.

Icon key and icon id specification using BNF

```
icon key ::= segment["."segment]*
segment ::= text+ [( "-" | "_" | ":" ) text]*
text ::== (char | number)+
char ::== A | B | ... | Z | a | b | ... | z
number ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
icon id ::= segment
```

Example icon mapping file

```
# Business Service Topology
bsm.business-service = business_service ①
bsm.ip-service = IP_service ②
bsm.reduction-key = reduction_key ③
```

- 1 Icon definition for Business Services
- ② Icon definition for IP Services
- 3 Icon definition for Reduction Keys

11.2.1. Icon resolution

The icon of a vertex is resolved as follows:

- If a vertex id to icon id mapping is defined, the icon referenced by the icon id is used
- If a mapping for the icon key determined by the *Topology Provider* for the vertex is defined, the icon referenced by the icon id is used
 - If no mapping exists and the icon key has more than one segments, reduce the icon key by the last segment and try resolving that icon key
- If no mapping is defined, the fallback icon key default is used.

The following example icon mapping is defined for the *Linkd Topology Provider* to illustrate this behaviour.

```
linkd.system.snmp.1.3.6.1.4.1.9.1.485 = server1
linkd.system.snmp.1.3.6 = server2
```

If the Enterprise OID of a node is 1.3.6.1.4.1.9.1.485 the icon with id server1 is used. If the Enterprise OID of a node is 1.3.6 the icon with id server2 is used. However, if the Enterprise OID of a node is 1.3.6.1.4.1.9.1.13 the icon with id server2 is used.

Linkd Topology Provider

The *Linkd Topology Provider* uses the **Enterprise OID** from each node to determine the icon of a vertex.

11.2.2. Change existing icon mappings

The easiest way to change an icon representation of an existing Vertex is to use the *Icon Selection Dialog* from the Vertex' context menu in the *Topology Map*. This will create a custom icon key to icon id mapping in the *Topology Provider* specific icon mapping file. As icon key the Vertex id is used. This allows each Vertex to have it's own icon.

If a more generic approach is preferred the icon mapping file can be modified manually.



Do NOT remove the default mappings and do NOT change the icon keys in the default mappings.

11.2.3. Add new icons

All available icons are stored in SVG files located in \${OPENNMS_HOME}/jetty-webapps/opennms/svg. To add new icons, either add definitions to an existing SVG file or create a new SVG file in that directory.

Whatever way new icons are added to *OpenNMS* it is important that each new icon id describes a set of icons, rather than a single icon. The following example illustrates this.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"</pre>
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg id="icons" xmlns="http://www.w3.org/2000/svg">
 <q id="my-custom_icon"> ①
     <q id="my-custom active"> ②
          <!-- rect, path, circle, etc elements, supported by SVG -->
     </g>
     <g id="my-custom_rollover"> ③
          <!-- rect, path, circle, etc elements, supported by SVG -->
     </g>
     <q id="my-custom"> 4
          <!-- rect, path, circle, etc elements, supported by SVG -->
 </q>
 <!-- Additional groups ... -->
</svg>
```

- ① Each icon must be in a SVG group with the id <icon id>_icon. Each SVG <icon id>_icon group must contain three sub groups with the ids: <icon id>_active, <icon id>_rollover and <icon id>.
- 2) The icon to use when the Vertex is selected.
- 3 The icon to use when the Vertex is moused over.
- ④ The icon to use when the Vertex is not selected or not moused over (just visible).
 - 0

It is important that each icon id is unique overall SVG files. This means there cannot be another my-custom icon id in any other SVG file.

If the new icons should be selectable from the *Topology Map's Icon Selection Dialog* an entry with the new icon id must be added to the file \${OPENNMS_HOME}/etc/org.opennms.features.topology.app.icons.properties.

Snippet of org.opennms.features.topology.app.icons.list

```
access_gateway ①
accesspoint
cloud
fileserver
linux_file_server
opennms_server
printer
router
workgroup_switch
my-custom ②
```

- 1 Already existing icon ids
- 2 New icon id



The order of the entries in org.opennms.features.topology.app.icons.list determine the order in the *Icon Selection Dialog* in the *Topology Map*.

Chapter 12. Asset Topology Provider

12.1. Overview

OpenNMS Meridian has introduced the ability for users to define arbitrarily complex layered topologies using GraphML (see http://graphml.graphdrawing.org/). The details of how OpenNMS Meridian interprets GraphML are given in the GraphML section of the OpenNMS Meridian developers guide. The ability to display complex layered topologies is a great feature but creating a usable GraphML topology for a large network can be a complex task for a user.

The Asset Topology Provider avoids the need for users to work directly with GraphML by directly generating a layered GraphML topology based upon node parameters and the contents of the Node Asset table. The Asset Topology Provider greatly simplifies the task for many use cases by allowing users to define fields in the Node Asset table which will enable nodes to be positioned correctly in a complex topology. This allows a physical and logical ordering of nodes which makes it easier for users to represent and navigate their infrastructure.

The structure of the generated topology is determined by the assetLayers configuration constant which can be set by a user. To illustrate how this works, we will consider the following configuration:

assetLayers=asset-region,asset-building

The *OpenNMS Meridian* Asset table is parsed to generate nested layers in the order of the comma separated keys in the assetLayers property. Each layer is a graph which is named after the key. Graph nodes in each layer reference related Graph nodes in the underlying layer. The lowest layer contains Graph nodes which are directly linked to monitored *OpenNMS Meridian* nodes which have entries in the Asset table.

The following diagram shows the structure of a topology generated by the above assetLayers property

asset-region: north asset-building:21 asset-building:22 Node 1 Node 2 Node 3 Node 4 Node 5 Node 6 No

Node 4

asset-building:22

unallocated-no

Nested Set Representa

asset-region: north

Node 2

set-building:21

de 1

ntation topology-layers=asset-region,asset-building asset-region: north unallocated-no sset-building:21 asset-building:22 Node 6 Noc ode 1 Node 2 Node 3 Node 4 Node 5 Node 6 No Node 5 Node 3 Node 4 Node 2 unallocated-no de 1 asset-building:22 set-building:21 Nested Set Representa

In this example the region asset fields for node 1,2,3,4 are set to north. All of these nodes are in the same north region. The building asset fields for Node 1 and Node 2 are set to 21 (both nodes are in building 21) while the building asset fields for Node 3 and Node 4 are set to 22 (both nodes are in building 22).

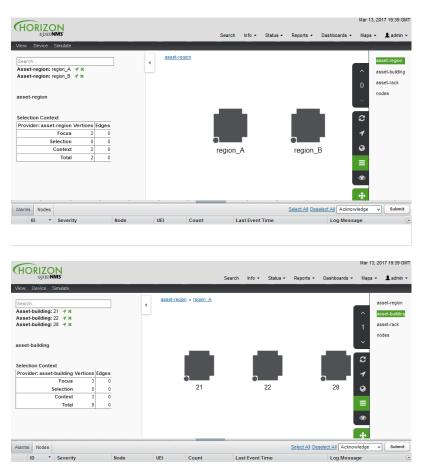
asset-region: north

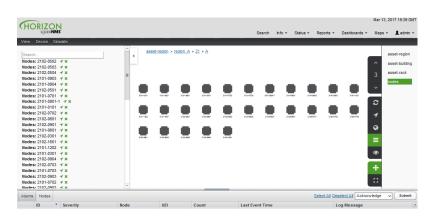
The Asset Topology Provider generates four linked graphs for this configuration. The layer 0 graph is called asset-region, the layer 1 graph is called asset-building and the layer 2 graph is called nodes.

Conceptually we can see that the topology is rendered as concentric sets. The *Asset Topology Provider* first searches all of the nodes with regions defined and creates a new level 0 graph node representing each region found. The *Asset Topology Provider* then searches within each region to find the building entries and creates a corresponding level 1 graph node for each building name found. Finally the *Asset Topology Provider* creates layer 2 nodes corresponding to each *OpenNMS Meridian* monitored node and places each in the correct building.

If however *OpenNMS Meridian* monitored nodes are found which have either the region or building asset fields empty they cannot be placed correctly in this topology. These nodes as shown in the diagram as unallocated nodes. Finally, only building and region nodes are generated which can be linked to *OpenNMS Meridian* nodes in the topology. The *Asset Topology Provider* does not generate spurious graph nodes in upper layers which are not directly and completely referenced by *OpenNMS Meridian* nodes in the lowest layer.

Example screenshots of a topology containing regions, buildings, racks and nodes are shown below





12.2. Asset layers

The entries for assetLayers can be any node or asset entry from the following list (defined in class NodeParamLabels). Keys beginning with node- come from the node table. Keys beginning with parent- come from the node table entry of the designated parent node (If defined). Keys beginning with asset- come from the corresponding asset table entry for the given node (If defined).

| node-nodelabel | node-nodeid | node-foreignsource | node-foreignid | node- nodesysname |
|--------------------------|-----------------------------|-----------------------------|-------------------------------------|--------------------------|
| node- nodesyslocation | node- operatingsystem | node-categories | | |
| parent-nodelabel | parent-nodeid | parent- foreignsource | parent-foreignid | |
| asset-address1 | asset-address2 | asset-city | asset-zip | asset-state |
| asset-latitude | asset-longitude | asset-region | asset-division | asset- department |
| asset-building | asset-floor | asset-room | asset-rack | asset-slot |
| asset-port | asset-circuitid | asset-category | asset-displaycategory | asset- notifycategory |
| asset- pollercategory | asset- thresholdcategory | asset- managedobjecttype | asset- managedobjectinstanc e | asset- manufacturer |
| asset-vendor | asset- modelnumber | asset-description | asset-operatingsystem | asset-country |

This allows arbitrary topologies to be generated including physical fields (room, rack etc.) and logical fields such as asset node categories. Please note you should not put any spaces in the comma separated assetLayers list. If the assetLayers property is defined as empty then a single graph layer will be generated containing all opennms nodes.

12.3. Node filtering

In many cases it is desirable to control which nodes are included or excluded from a topology. For instance it is useful to be able to generate customised topologies for specific customers which include only regions/buildings etc relevant to their filtered node set. To this end it is possible to define a node filter which chooses which nodes are included in a generated topology.

Filters are defined using the same asset table keys which are available for the assetLayers field.

| Operatio n | Definition | Example |
|---------------|---|--------------------------|
| | key1=value1,value2 alternatively key1=value1;key1=value2 | asset-region=north,south |

| Operatio n | Definition | Example |
|---------------|------------|--|
| AND | | asset-region=north;asset- building=23 |
| NOT | key1=!val1 | asset-building=!23 |

Thus the following configuration means include only nodes with region north or south but exclude all nodes with building 23.

```
filter=asset-region=north,south;asset-building=!23
```

The filters are designed to treat all selected text key entries as comma separated values (csv). This allows OpenNMS node-categories which are many to many entries to be dealt with as a comma separated list of values; routers, servers, web etc. Thus we can select based on multiple separate node categories. The following configuration means show routers and servers on all buildings except building 23.

```
filter=node-categories=routers,servers;asset-building=!23
```

The filters treat all asset table entries as comma separated variables (csv). This also means that, for instance asset-displaycategory could also contain several values separated by commas. e.g. customer1,customer2,customer3 etc.



You should make sure asset addresses and other free format asset text fields do not contain commas if you want an exact match on the whole field

Regular expressions are also allowed. Regular expressions start with the \sim character. You can also negate a regular expression by preceding it with ! \sim .

The following example will match against regions 'Stuttgart' and 'Isengard' and any building name which ends in 4

```
filter=asset-region=~.*gar(t|d);asset-building=~.*4
```

12.4. Configuration

The *Asset Topology Provider* persists both the asset topology graph definitions and the generated GraphML graphs. The persisted definitions mean that is is possible to regenerate graphs if the asset table is changed without reentering the configuration.

The Asset Topology Provider persists GraphML graphs along side any other GraphML graphs in the directory;

```
<opennms home>/etc/graphml
```

Please note that if you are using ReST or any other means to generate other GraphML graphs, you should ensure that the providerIds and labels are distinct from those used by the *Asset Topology Provider*

The asset graph definitions for the Asset Topology Provider are persisted to the following xml configuration file:

```
<opennms home>/etc/org.opennms.features.topology.plugins.topo.asset.xml
```

Normally you should not edit this file directly but use the karaf consol or events to define new graphs.

The config file will contain each of the graph definitions as properties in the form

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<configs>
    <config>
        <label>Asset Topology Provider</label>
        <breadcrumb-strategy>SHORTEST_PATH_TO_ROOT</breadcrumb-strategy>
        ovider-id>asset/provider-id>
        <preferred-layout>Grid Layout</preferred-layout>
        <filters>
            <filter>asset-region=South</filter>
        </filters>
        <layers>
            <layer>asset-region</layer>
            <layer>asset-building</layer>
            <layer>asset-rack</layer>
        </layers>
    </config>
</configs>
```

The individual definition parameters are described in the following table

| Parameter | Description |
|-----------------|---|
| providerId | The unique name of the provider - used as handle to install and remove the topology |
| label | The name which shows up on the topology menu (must be unique) |
| assetLayers | List of asset layers (in order). See separate description. |
| filters | List of filters to be applied. Filters determine which nodes are included in graph. See separate description. |
| preferredLayout | Preferred layout of the nodes in generated graphs. |

| Parameter | Description |
|------------------------|--|
| breadcrumbStrat egy | Breadcrumb strategy used to display breadcrumbs above each graph |

12.5. Creating Asset Based Topologies From Karaf Consol

The *OpenNMS Meridian* Karaf Consol can be used to control topology generation. To login use admin password.

ssh admin@localhost -p 8101

The following commands are available

| Command | Description | Options |
|--------------------------------------|--|---|
| asset- topology:cre ate | Creates Asset Topology. | (The default settings are used if a particular setting is not included in the command) -l,label: Asset Topology label (shows in topology menu) (Default: asset) -i,providerId: Unique providerId of asset topology (Default: 'Asset Topology Provider') -f,filter: Optional node filter (Default: empty filter i.e. allow all nodes) -a,assetLayers: Comma separated list of asset layers (Default: asset-region, asset-building, asset-rack) -p,preferredLayout: Preferred Layout (Default: 'Grid Layout') -b,breadcrumbStrategy: Bread Crumb Strategy (Default: SHORTEST_PATH_TO_ROOT) If you simply type asset-topology:create a default topology with providerId asset will be created. |
| asset- topology:re move | Removes Asset Topology. | -i,providerId : Unique providerId of asset topology (Default: asset) |
| asset- topology:list | Lists all Asset Topologies installed. | all : display detailed view includinguriParams string |
| asset- topology:reg enerate | Regenerates the graphs for the given Asset Topology definition. | -i,providerId : Unique providerId of asset topology to regenerate (Default: asset) |
| asset- topology:reg enerateall | Best Effort regeneration of all asset topologies. (If one graph fails, the command will try to complete the rest of the definitions definition) | |

12.6. Creating Asset Based Topologies Using *OpenNMS Meridian* **events**

The Asset Topology Provider listens for events which trigger the generation and installation or removal of topologies. The Asset Topology Provider events are defined in the file

```
<opennms home>/etc/events/GraphMLAssetPluginEvents.xml
```

These events will use the default parameters if parameters are not supplied

To create a new topology from the current OpenNMS inventory use

```
(for default topology)
sudo ./send-event.pl uei.opennms.plugins/assettopology/create localhost

(or with parameters)
sudo ./send-event.pl uei.opennms.plugins/assettopology/create localhost -p
'providerId test' -p 'label test' -p 'assetLayers asset-country,asset-city,asset-building'-->

other example possible parameters are
-p 'filters asset-displaycategory=!testDisplayCategory'
-p 'preferredLayout Grid Layout'
-p 'breadcrumbStrategy SHORTEST_PATH_TO_ROOT'
```

To uninstall an asset topology use

```
(for default topology providerId)
sudo ./send-event.pl uei.opennms.plugins/assettopology/remove localhost

(or with specific providerId)
sudo ./send-event.pl uei.opennms.plugins/assettopology/remove localhost -p
'providerId test'
```

To regenerate an existing asset topology use

```
(for default topology providerId)
sudo ./send-event.pl uei.opennms.plugins/assettopology/regenerate localhost

(or with specific providerId)
sudo ./send-event.pl uei.opennms.plugins/assettopology/regenerate localhost-p
'providerId test'
```

To regenerate all existing asset topologies use

12.7. Viewing the topology

If all goes well, having installed the topology, upon refreshing your screen, you should see a new topology display option in the *OpenNMS Meridian* topology page. The displayed name of this topology is given by the label field

The label field need not be the same as the providerId which is used by the ReST api for the installation or removal of a topology. However the label field must be unique across all installed topologies.

It is possible to have several topologies installed which have been generated using different configurations. You simply need to ensure that the providerId and label field used for each installation command is different.

12.8. additional notes

Please note you MUST first uninstall an *OpenNMS Meridian* graphml topology before installing a new one. You will also have to log out and log back into the UI in order to see the new topology file. If you uninstall a topology while viewing it, the UI will throw an error and you will also have to log out and back in to see the remaining topologies.

Chapter 13. Database Reports

Reporting on information from the *OpenNMS Meridian* monitoring system is important for strategical or operational decisions. *Database Reports* give access to the embedded *JasperReports* engine and allows to create and customize report templates. These reports can be executed on demand or on a pre-defined schedule within *OpenNMS Meridian*.



Originally *Database Reports* were introduced to create reports working on data stored in the *OpenNMS Meridian* database only. This is no longer mandatory, also performance data can be used. Theoretically the reports do not necessarily need to be *OpenNMS Meridian* related.



The *OpenNMS Meridian Report Engine* allows the creation of various kinds of reports and also supports distributed report repositories. At the moment these features are not covered by this documentation. Only reports using *JasperReports* are described here.

13.1. Overview

The *OpenNMS Meridian Report Engine* uses the *JasperReport* library to create reports in various output formats. Each report template must be a *.jrxml file. The *OpenNMS Meridian Report Engine* passes a *JDBC* Connection to the *OpenNMS Meridian Database* to each report on execution.

Table 85. feature overview

| Supported Output Formats | PDF, CSV |
|--------------------------|----------|
| JasperReport Version | 6.3.0 |

For more details on how *JasperReports* works, please have a look at the official documentation of *Jaspersoft Studio*.

13.2. Modify existing reports

All default reports of *OpenNMS Meridian* are located in <code>\$OPENNMS_HOME/etc/report-templates</code>. Each <code>.jrxml</code> file located there can be modified and the changes are applied the next time a report is created by <code>OpenNMS Meridian</code>.

When a subreport has been modified *OpenNMS Meridian* will detect a change based on the report's lastModified time and will recompile the report. A compiled version of the report is represented by a .jasper file with the same name as the .jrxml file. Subreports are located in \$OPENNMS_HOME/etc/report-templates/subreports.



If unsure, simply delete all .jasper files and *OpenNMS Meridian* will automatically compile the subreports if needed.

13.3. Add a custom report

To add a new JasperReport report to the Local OpenNMS Meridian Report Repository, the following steps are required.

At first a new entry in the file \$OPENNMS_HOME/etc/database-reports.xml must be created.

```
<report
  id="MyReport" ①
  display-name="My Report" ②
  online="true" ③
  report-service="jasperReportService" ④
  description="This is an example description. It shows up in the web ui when creating an online report" ⑤
/>
```

- 1 A unique identifier.
- 2 The name of the report. Is shown when using the web ui.
- 3 Defines if this report can be executed on demand, otherwise only scheduling is possible.
- 4 The report service implementation to use. In most cases this is jasperReportService.
- ⑤ A description of the report. Is shown when using the web ui.

In addition a new entry in the file \$OPENNMS_HOME/etc/jasper-reports.xml must be created.

```
<report
  id="MyReport" ①
  template="My-Report.jrxml" ②
  engine="jdbc" ③
/>
```

- ① The identifier defined in the previous step. This identifier must exist in \$OPENNMS_HOME/etc/database-reports.xml.
- ② The name of the template. The template must be located in \$OPENNMS_HOME/etc/report-templates.
- 3 The engine to use. It is either jdbc or null.

13.4. Usage of Jaspersoft Studio

When developing new reports it is recommended to use the *Jaspersoft Studio* application. It can be downloaded here.



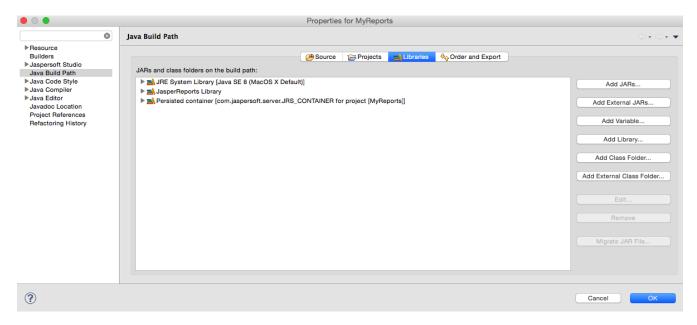
We recommend always to use the same *Jaspersoft Studio* version as the *JasperReport* library OpenNMS Meridian uses. Currently OpenNMS Meridian uses version 6.3.0.

13.4.1. Connect to the OpenNMS Meridian Database

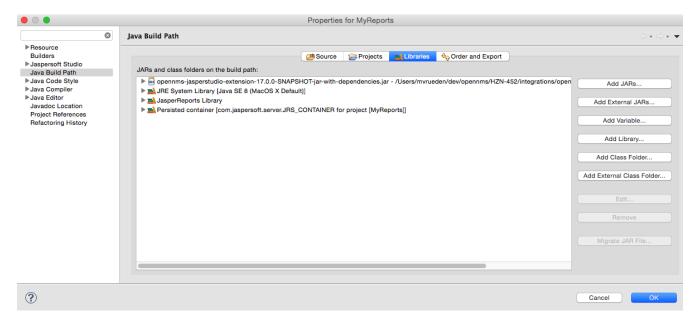
In order to actually create SQL statements against the *OpenNMS Meridian database* a database Data Adapter must be created. The official *Jaspersoft Studio* documentation and wiki covers this aspect.

13.4.2. Use Measurements Datasource and Helpers

To use the *Measurements API* it is required to add the *Measurements Datasource* library to the build path of *JasperStudio*. This is achieved with right click in the Project Explorer and select Configure Buildpath.

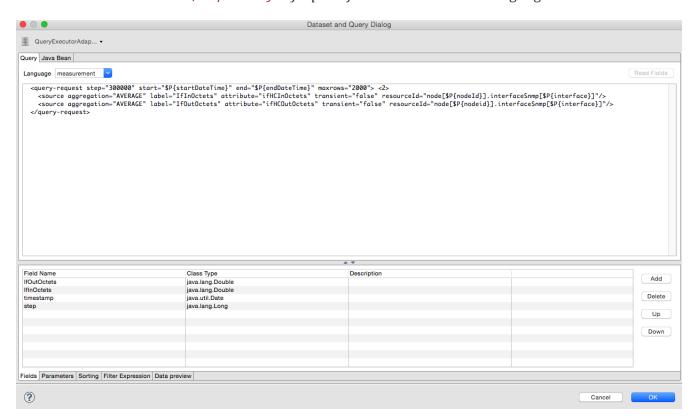


- 1. Switch to the Libraries tab.
- 2. Click Add External JARs and select the opennms-jasperstudio-extension-2017.1.6-jar-with-dependencies.jar file located in \$OPENNMS_HOME/contrib/jasperstudio-extension.
- 3. Close the file selection dialog.



- 1. Close the dialog.
- 2. The *Measurements Datasource and Helpers* should now be available.

3. Go to the Dataset and Query Dialog in Jaspersoft Studio and select a language called measurement.





Even if there is no Read Fields functionality available, the Data preview can be used. It is required the the access to the *Measurements API* is possible using the connection parameters MEASUREMENT_URL, MEASUREMENT_USERNAME and MEASUREMENT_PASSWORD. The Supported Fields section gives more details. In addition you have

13.5. Accessing Performance Data



Before *OpenNMS Horizon 17* and *OpenNMS Meridian 2016* it was possible to access the performance data stored in .rrd or .jrobin files directly by using the jrobin language extension provided by the RrdDataSource. This is no longer possible and the *Measurements Datasource* has to be used.

To access performance data within reports we created a custom *Measurement Datasource* which allows to query the *Measurements API* and process the returned data in your reports. Please refer to the official Measurements API documentation on how to use the Measurements API.



When using the *Measurements Datasource* within a report a HTTP connection to the *Measurements API* is only established if the report is NOT running within OpenNMS Meridian, e.g. when used with *Jaspersoft Studio*.

To receive data from the *Measurements API* simply create a query as follows:

Sample queryString to receive data from the Measurements API

① The query language. In our case measurement, but JasperReports supports a lot out of the box, such as sql, xpath, etc.

13.5.1. Fields

Each datasource should return a number of fields, which then can be used in the report. The *Measurement Datasource* supports the following fields:

| Field name | Field type | Field description |
|-----------------|------------------|---|
| <label></label> | java.lang.Double | Each Source defined as transient=false can be used as a field. The name of the field is the label, e.g. IfInOctets |
| timestamp | java.util.Date | The timestamp of the sample. |
| step | java.lang.Long | The Step size of the Response. Returns the same value for all rows. |
| start | java.lang.Long | The Start timestamp in milliseconds of the Resonnse. Returns the same value for all rows. |
| end | java.lang.Long | The End timestamp in milliseconds of the Response. Returns the same value for all rows. |

For more details about the Response, please refer to the official Measurement API documentation.

13.5.2. Parameters

In addition to the queryString the following *JasperReports* parameters are supported.

| Parameter name | Required | Description |
|----------------------|----------|--|
| MEASUREMENT_URL | yes | The URL of the <i>Measurements API</i> , e.g. http://localhost:8980/opennms/ rest/measurements |
| MEASUREMENT_USERNAME | no | If authentication is required, specify the username, e.g. admin |
| MEASUREMENT_PASSWORD | no | If authentication is required, specify the password, e.g. admin |

13.6. Helper methods

There are a couple of helper methods to help creating reports in *OpenNMS Meridian*.

These helpers come along with the *Measurement Datasource*.

Table 86. supported helper methods

| Helper class | Helper Method | Description |
|---|--|---|
| org.opennms.netmgt.jasper.help er.MeasurementsHelper | getNodeOrNodeSourceDescriptor(nodeId, foreignSource, foreignId) | Generates a node source descriptor according to the input paramters. Either node[nodeId] or nodeSource[foreignSource:forei gnId] is returned. nodeSource[foreignSource:forei gnId] is only returned if foreignSource and foreignId is not empty and not null. Otherwise always node[nodeId] is returned. nodeId: String, the id of the node foreignSource: String, the foreign source of the node, may be null foreignId: String, the foreign id of the node, may be null. For more details checkout Usage of the node source descriptor. |

13.6.1. Usage of the interface descriptor

An interfaceSnmp is addressed with the exact interface descriptor. To allow easy access to the interface descriptor a helper tool is provided. The following example shows the usage of that helper.

jrxml report snippet to visualize the use of the interface descriptor

13.6.2. Usage of the node source descriptor

A node is addressed by a node source descriptor. The node source descriptor references the node either via the foreign source and foreign id or by the node id.

If store by foreign source is enabled only addressing the node via foreign source and foreign id is possible.

In order to make report creation easier, there is a helper method to create the node source descriptor.



For more information about store by foreign source, please have a look at our Wiki.

The following example shows the usage of that helper.

jrxml report snippet to visualize the use of the node source descriptor.

```
<parameter name="nodeResourceDescriptor" class="java.lang.String" isForPrompting=</pre>
"false">
  <defaultValueExpression>
<![CDATA[org.opennms.netmgt.jasper.helper.MeasurementsHelper.getNodeOrNodeSourceDescri</pre>
ptor(String.valueOf($P{nodeid}), $P{foreignsource}, $P{foreignid})
1]></defaultValueExpression>
</parameter>
<queryString language="Measurement">
  <![CDATA[<query-request step="300000" start="$P{startDateTime}"</pre>
end="$P{endDateTime}" maxrows="2000">
<source aggregation="AVERAGE" label="IfInOctets" attribute="ifHCInOctets"</pre>
transient="false" resourceId="$P{nodeResourceDescriptor}.interfaceSnmp[en0-
005e607e9e001"/>
<source aggregation="AVERAGE" label="IfOutOctets" attribute="ifHCOutOctets"</pre>
transient="false" resourceId="$P{nodeResourceDescriptor}.interfaceSnmp[en0-
005e607e9e001"/>
</query-request>]]>
```

Depending on the input parameters you either get a node resource descriptor or a foreign source/foreign id resource descriptor.

13.6.3. Usage of the interface descriptor

An interfaceSnmp is addressed with the exact interface descriptor. To allow easy access to the interface descriptor a helper tool is provided. The following example shows the usage of that helper.

jrxml report snippet to visualize the use of the interface descriptor

To get the appropriate interface descriptor depends on the input parameter.

13.6.4. Use HTTPS

To establish a secure connection to the *Measurements API* the public certificate of the running *OpenNMS Meridian* must be imported to the *Java Trust Store*. In Addition *OpenNMS Meridian* must be configured to use that *Java Trust Store*. Please follow the instructions in this chapter to setup the *Java Trust Store* correctly.

In addition please also set the property org.opennms.netmgt.jasper.measurement.ssl.enable in \$OPENNMS_HOME\etc\opennms.properties to true to ensure that only secure connections are established.



If org.opennms.netmgt.jasper.measurement.ssl.enable is set to false an accidentally insecure connection can be established to the *Measurements API* location. A SSL secured connection can be established even if org.opennms.netmgt.jasper.measurement.ssl.enable is set to false.

13.7. Limitations

• Only a *JDBC Datasource* to the *OpenNMS Meridian Database connection* can be passed to a report, or no datasource at all. One does not have to use the datasource, though.

Chapter 14. Enhanced Linkd

Enhanced Linkd (Enlinkd) has been designed to discover connections between nodes using data generated by various link discovery protocols and accessible via SNMP. Enlinkd gathers this data on a regular interval and creates a snapshot of a device's neighbors from its perspective. The connections discovered by Enlinkd are called Links. The term Link, within the context of Enlinkd, is not synonymous with the term "link" when used with respect to the network OSI Layer 2 domain, whereby a link only indicates a Layer 2 connection. A Link in context of Enlinkd is a more abstract concept and is used to describe any connection between two OpenNMS Meridian Nodes. These Links are discovered based on information provided by an agent's understanding of connections at the OSI Layer 2, Layer 3, or other OSI layers.

The following sections describe the *Enlinkd* daemon and its configuration. Additionally, the supported *Link discovery* implementations will be described as well as a list of the SNMP MIBs that the SNMP agents must expose in order for *EnLinkd* to gather *Links* between *Nodes*. FYI: Detailed information about a node's connections (discovered *Links*) and supporting link data can be seen on the *Node detail page* within the *OpenNMS Meridian* Web-UI.

14.1. Enlinkd Daemon

Essentially *Enlinkd* asks each device the following question: "What is the network topology from your point of view". From this point of view this will only provide local topology discovery features. It does not attempt to discover global topology or to do any correlation with the data coming from other nodes.

For large environments the behavior of *Enlinkd* can be configured. During the *Link* discovery process informational and error output is logged to a global log file.

Table 87. Global log and configuration files for Enlinkd

| File | Location | Description |
|---------------------------|--------------------------------|--|
| enlinkd-configuration.xml | <pre>\$OPENNMS_HOME/etc</pre> | Global configuration for the daemon process |
| enlinkd.log | <pre>\$OPENNMS_HOME/logs</pre> | Global <i>Enlinkd</i> log file |
| log4j2.xml | <pre>\$OPENNMS_HOME/etc</pre> | Configuration file to set the log level for <i>Enlinkd</i> |

Table 88. Descriptione for global configuration parameter

| Attribute | Туре | Defau lt | Description |
|--------------------------|-------------|--------------|--|
| threads | Intege r | 5 | Number of parallel threads used to discover the topology. |
| initial_sleep_ time | Intege r | 60000 | Time in milliseconds to wait for discovering the topology after OpenNMS Meridian is started. |
| rescan_interva l | Intege r | 864000 00 | Interval to rediscover and update the topology in milliseconds. |
| use-cdp- discovery | Boole an | true | Enable or disable topology discovery based on <i>CDP</i> information. |
| use-bridge- discovery | Boole an | true | Enable or disable algorithm to discover the topology based on the <i>Bridge MIB</i> information. |
| use-lldp- discovery | Boole an | true | Enable or disable topology discovery based on <i>LLDP</i> information. |
| use-ospf- discovery | Boole an | true | Enable or disable topology discovery based on <i>OSPF</i> information. |
| use-isis- discovery | Boole an | true | Enable or disable topology discovery based on <i>IS-IS</i> information. |



If multiple protocols are enabled, the links will be discovered for each enabled discovery protocol. The topology WebUI will visualize *Links* for each discovery protocol. For example if you start *CDP* and *LLDP* discovery, the WebUI will visualize a *CDP Link* and an *LLDP Link*.

14.2. Layer 2 Link Discovery

Enlinkd is able to discover *Layer 2* network links based on the following protocols:

• Link Layer Discovery Protocol (LLDP)

- Cisco Discovery Protocol (CDP)
- Transparent Bridge Discovery

This information are provided by *SNMP Agents* with appropriate *MIB support*. For this reason it is required to have a working *SNMP* configuration running. The following section describes the required *SNMP MIB* provided by the *SNMP agent* to allow the *Link Discovery*.

14.2.1. LLDP Discovery

The Link Layer Discovery Protocol (LLDP) is a vendor-neutral link layer protocol. It is used by network devices for advertising their identity, capabilities, and neighbors. LLDP performs functions similar to several proprietary protocols, such as the Cisco Discovery Protocol (CDP), Extreme Discovery Protocol, Foundry Discovery Protocol (FDP), Nortel Discovery Protocol (also known as SONMP), and Microsoft's Link Layer Topology Discovery (LLTD) [1: Wikipedia LLDP: https://en.wikipedia.org/wiki/Link_Layer_Discovery_Protocol].



Only nodes with a running *LLDP* process can be part of the link discovery. The data is similar to running a show lldp neighbor command on the device. Linux and Windows servers don't have an *LLDP* process running by default and will not be part of the link discovery.

The following OIDs are supported to discover and build the *LLDP* network topology.

Table 89. Supported OIDs from LLDP-MIB

| Name | OID | Description |
|-----------------------------|-------------------------------|--|
| lldpLocChassisI dSubtype | .1.0.8802.1.1.2. 1.3.1.0 | The type of encoding used to identify the chassis associated with the local system. Possible values can be: chassisComponent(1) interfaceAlias(2) portComponent(3) macAddress(4) networkAddress(5) interfaceName(6) local(7) |
| lldpLocChassisI d | .1.0.8802.1.1.2. 1.3.2.0 | The string value used to identify the chassis component associated with the local system. |
| lldpLocSysName | .1.0.8802.1.1.2. 1.3.3.0 | The string value used to identify the system name of the local system. If the local agent supports IETF RFC 3418, lldpLocSysName object should have the same value of sysName object. |
| lldpLocPortIdSu btype | .1.0.8802.1.1.2. 1.3.7.1.2 | The type of port identifier encoding used in the associated <i>lldpLocPortId</i> object. |
| lldpLocPortId | .1.0.8802.1.1.2. 1.3.7.1.3 | The string value used to identify the port component associated with a given port in the local system. |

| Name | OID | Description |
|-----------------------------|-------------------------------|--|
| lldpLocPortDesc | .1.0.8802.1.1.2. 1.3.7.1.4 | The string value used to identify the 802 LAN station's port description associated with the local system. If the local agent supports IETF RFC 2863, lldpLocPortDesc object should have the same value of ifDescr object. |
| lldpRemChassisI dSubtype | .1.0.8802.1.1.2. 1.4.1.1.4 | The type of encoding used to identify the chassis associated with the local system. Possible values can be: chassisComponent(1) interfaceAlias(2) portComponent(3) macAddress(4) networkAddress(5) interfaceName(6) local(7) |
| lldpRemChassisI d | .1.0.8802.1.1.2. 1.4.1.1.5 | The string value used to identify the chassis component associated with the remote system. |

| Name | OID | Description |
|--------------------------|-------------------------------|--|
| lldpRemPortIdS ubtype | .1.0.8802.1.1.2.1.4.1.1.6 | The type of port identifier encoding used in the associated <code>lldpRemPortId</code> object. <code>interfaceAlias(1)</code> the octet string identifies a particular instance of the <code>ifAlias</code> object (defined in IETF RFC 2863). If the particular <code>ifAlias</code> object does not contain any values, another port identifier type should be used. <code>portComponent(2)</code> the octet string identifies a particular instance of the <code>entPhysicalAlias</code> object (defined in IETF RFC 2737) for a port or backplane component. <code>macAddress(3)</code> this string identifies a particular unicast source address (encoded in network byte order and IEEE 802.3 canonical bit order) associated with the port (IEEE Std 802-2001). <code>networkAddress(4)</code> this string identifies a network address associated with the port. The first octet contains the <code>IANA AddressFamilyNumbers</code> enumeration value for the specific address type, and octets 2 through N contain the <code>networkAddress</code> address value in network byte order. <code>interfaceName(5)</code> the octet string identifies a particular instance of the <code>ifName</code> object (defined in IETF RFC 2863). If the particular <code>ifName</code> object does not contain any values, another port identifier type should be used. <code>agentCircuitId(6)</code> this string identifies a agent-local identifier of the circuit (defined in RFC 3046) <code>local(7)</code> this string identifies a locally assigned port ID. |
| lldpRemPortId | .1.0.8802.1.1.2. 1.4.1.1.7 | The string value used to identify the port component associated with the remote system. |
| lldpRemPortDes c | .1.0.8802.1.1.2. 1.4.1.1.8 | The string value used to identify the description of the given port associated with the remote system. |
| lldpRemSysNam e | .1.0.8802.1.1.2. 1.4.1.1.9 | The string value used to identify the system name of the remote system. |
| | | |

Generic information about the *LLDP* process can be found in the *LLDP Information* box on the *Node Detail Page* of the device. Information gathered from these OIDs will be stored in the following database table:

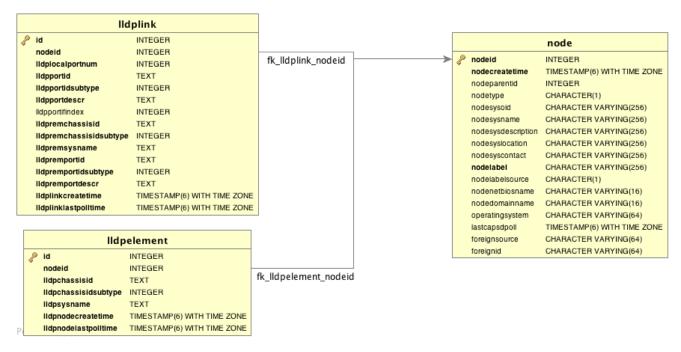


Figure 21. Database tables related to LLDP discovery

14.2.2. CDP Discovery

The *Cisco Discovery Protocol (CDP)* is a proprietary link layer protocol from *Cisco*. It is used by network devices to advertise identity, capabilities and neighbors. *CDP* performs functions similar to several proprietary protocols, such as the *Link Layer Discovery Protocol (LLDP)*, *Extreme Discovery Protocol*, *Foundry Discovery Protocol (FDP)*, *Nortel Discovery Protocol (also known as SONMP)*, and *Microsoft's Link Layer Topology Discovery (LLTD)*. The *CDP* discovery uses information provided by the CISCO-CDP-MIB and CISCO-VTP-MIB.



Only nodes with a running *CDP* process can be part of the link discovery. The data is similar to running a show cdp neighbor command on the IOS CLI of the device. Linux and Windows servers don't have a *CDP* process running by default and will not be part of the link discovery.

The following OIDs are supported to discover and build the CDP network topology.

Table 90. Supported OIDS from the IF-MIB

| Nam e | OID | Description |
|-------------|-----------------|--|
| ifDes cr | .1.3.6.1.2.1.2. | A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the interface hardware/software. |

Table 91. Supported OIDS from the CISCO-CDP-MIB to discover links

| Name | OID | Description |
|----------------------|--------------------------------|---|
| cdpInterfaceNa me | .1.3.6.1.4.1.9.9.2 3.1.1.1.1.6 | The name of the local interface as advertised by <i>CDP</i> in the <i>Port-ID TLV</i> . |

| Name | OID | Description |
|-------------------------|---------------------------------|--|
| cdpCacheEntry | .1.3.6.1.4.1.9.9.2 3.1.2.1.1 | An entry (conceptual row) in the <i>cdpCacheTable</i> , containing the information received via <i>CDP</i> on one interface from one device. Entries appear when a <i>CDP</i> advertisement is received from a neighbor device. Entries disappear when <i>CDP</i> is disabled on the interface, or globally. |
| cdpCacheAddres sType | .1.3.6.1.4.1.9.9.2 3.1.2.1.1.3 | An indication of the type of address contained in the corresponding instance of <i>cdpCacheAddress</i> . |
| cdpCacheAddres s | .1.3.6.1.4.1.9.9.2 3.1.2.1.1.4 | The (first) network-layer address of the device's SNMP-agent as reported in the Address <i>TLV</i> of the most recently received <i>CDP</i> message. For example, if the corresponding instance of <i>cacheAddressType</i> had the value <code>ip(1)</code> , then this object would be an IP-address. |
| cdpCacheVersio n | .1.3.6.1.4.1.9.9.2 3.1.2.1.1.5 | The Version string as reported in the most recent <i>CDP</i> message. The zero-length string indicates no Version field <i>(TLV)</i> was reported in the most recent <i>CDP</i> message. |
| cdpCacheDevice Id | .1.3.6.1.4.1.9.9.2 3.1.2.1.1.6 | The <i>Device-ID</i> string as reported in the most recent <i>CDP</i> message. The zero-length string indicates no <i>Device-ID</i> field (<i>TLV</i>) was reported in the most recent <i>CDP</i> message. |
| cdpCacheDevice Port | .1.3.6.1.4.1.9.9.2 3.1.2.1.1.7 | The <i>Port-ID</i> string as reported in the most recent <i>CDP</i> message. This will typically be the value of the <i>ifName</i> object (e.g., Ethernet0). The zero-length string indicates no <i>Port-ID</i> field (<i>TLV</i>) was reported in the most recent <i>CDP</i> message. |
| cdpCachePlatfor m | .1.3.6.1.4.1.9.9.2 3.1.2.1.1.8 | The Device's Hardware Platform as reported in the most recent <i>CDP</i> message. The zero-length string indicates that no Platform field <i>(TLV)</i> was reported in the most recent <i>CDP</i> message. |
| cdpGlobalRun | .1.3.6.1.4.1.9.9.2 3.1.3.1.0 | An indication of whether the Cisco Discovery Protocol is currently running. Entries in <i>cdpCacheTable</i> are deleted when CDP is disabled. |
| cdpGlobalDevice Id | .1.3.6.1.4.1.9.9.2 3.1.3.4.0 | The device ID advertised by this device. The format of this device id is characterized by the value of cdpGlobalDeviceIdFormat object. |

| Name | OID | Description |
|-----------------------------|--------------------|--|
| cdpGlobalDevice IdFormat | .1.3.6.1.4.1.9.9.2 | An indication of the format of Device-Id contained in the corresponding instance of <code>cdpGlobalDeviceId</code> . User can only specify the formats that the device is capable of as denoted in <code>cdpGlobalDeviceIdFormatCpb</code> object. <code>serialNumber(1):</code> indicates that the value of <code>cdpGlobalDeviceId</code> object is in the form of an <code>ASCII</code> string contain the device serial number. <code>macAddress(2):</code> indicates that the value of <code>cdpGlobalDeviceId</code> object is in the form of Layer 2 MAC address. <code>other(3):</code> indicates that the value of <code>cdpGlobalDeviceId</code> object is in the form specific <code>ASCII</code> string contain info that identifies the device. For example: <code>ASCII string</code> contains <code>serialNumber</code> appended/prepened with system name. |

Table 92. Supported OIDS from the CISCO-VTP-MIB.

| vtpVersion | .1.3.6.1.4.1.9.9.46 | The version of VTP in use on the local system. A device will report its version capability and not any particular version in use on the device. If the device does not support VTP, the version is none(3). |
|-----------------------|--------------------------------|---|
| ciscoVtpVla nState | .1.3.6.1.4.1.9.9.46 .1.3.1.1.2 | The state of this <i>VLAN</i> . The state <i>mtuTooBigForDevice</i> indicates that this device cannot participate in this <i>VLAN</i> because the <i>VLAN's MTU</i> is larger than the device can support. The state <i>mtuTooBigForTrunk</i> indicates that while this <i>VLAN's MTU</i> is supported by this device, it is too large for one or more of the device's trunk ports. <i>operational(1)</i> , <i>suspended(2)</i> , <i>mtuTooBigForDevice(3)</i> , <i>mtuTooBigForTrunk(4)</i> |
| ciscoVtpVla nType | .1.3.6.1.4.1.9.9.46 .1.3.1.1.3 | The type of this VLAN. ethernet(1), fddi(2), tokenRing(3), fddiNet(4), trNet(5), deprecated(6) |
| ciscoVtpVla nName | .1.3.6.1.4.1.9.9.46 .1.3.1.1.4 | The name of this <i>VLAN</i> . This name is used as the <i>ELAN-name</i> for an <i>ATM LAN-Emulation</i> segment of this <i>VLAN</i> . |

Generic information about the *CDP* process can be found in the *CDP Information* box on the *Node Detail Page* of the device. Information gathered from these OIDs will be stored in the following database table:

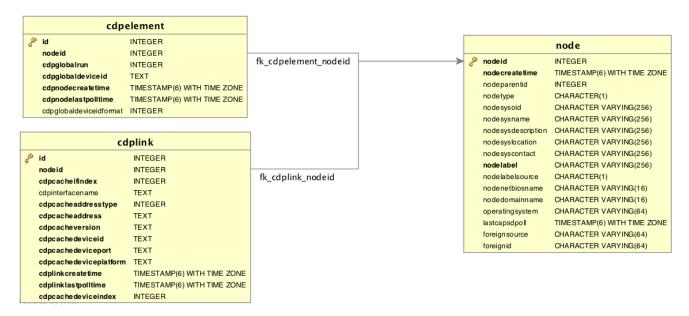


Figure 22. Database tables related to CDP discovery

14.2.3. Transparent Bridge Discovery

Discovering *Layer 2* network links using the *Bridge Forwarding* table requires a special algorithm. To discover *Links* an algorithm based on a scientific paper with the title Topology Discovery for Large Ethernet Networks is implemented. The gathered information is used to classify *Links* in *macLink* and *bridgeLink*. A *macLink* represents a *Link* between a workstation or server identified by a mac address. A *bridgeLink* is a *connection* between backbone ports.

Transparent bridging is not loop free so if you have loops you have to enable the spanning tree protocol that will detect loops and again will put some ports in a *blocking* state to avoid loops. To get links it is necessary to perform some calculations that let us define the *Links*. The following *MIBS* must be supported by the *SNMP agent* to allow *Transparent Bridge Discovery*.

Table 93. Supported MIBS from the Cisco-VTP MIB

| Nar | me | OID | Description | |
|---------|------|---------------------------------|---|----|
| vtp\ion | Vers | .1.3.6.1.4.1.9.9.46 .1.1.1.0 | The version of VTP in use on the local system. A device will report its version capability and not any particular version in use on the device. the device does not support <i>VTP</i> , the version is <i>none(3)</i> . | If |

Table 94. Supported OIDs from the IP-MIB

| Name | OID | Description |
|-----------------------------|----------------|---|
| ipNetToMediaIfIn dex | .1.3.6.1.2.1.4 | The interface on which this entry's equivalence is effective. The layer-2 interface identified by a particular value of this index is the same interface as identified by the same value of <i>ifIndex</i> . |
| ipNetToMediaPhy sAddress | .1.3.6.1.2.1.4 | The media-dependent <i>physical</i> address. |

| ipNetToMediaNet Address | .1.3.6.1.2.1.4 | The <i>IpAddress</i> corresponding to the media-dependent <i>physical</i> address. |
|----------------------------|----------------|--|
| ipNetToMediaTyp e | .1.3.6.1.2.1.4 | The type of mapping. Setting this object to the value <code>invalid(2)</code> has the effect of invalidating the corresponding entry in the <code>ipNetToMediaTable</code> . That is, it effectively dissasociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant <code>ipNetToMediaType</code> object. |

Table 95. Supported OIDS from the BRIDGE-MIB

| Name | OID | Description |
|-----------------------------------|-----------------|---|
| dot1dBaseBridgeA ddress | .1.3.6.1.2.1.17 | The MAC address used by this bridge when it must be referred to in a unique fashion. It is recommended that this be the numerically smallest MAC address of all ports that belong to this bridge. However it is only required to be unique. When concatenated with dot1dStpPriority a unique BridgeIdentifier is formed which is used in the Spanning Tree Protocol. |
| dot1dBaseNumPort s | .1.3.6.1.2.1.17 | The number of ports controlled by this bridging entity. |
| dot1dBaseType | .1.3.6.1.2.1.17 | Indicates what type of bridging this bridge can perform. If a bridge is actually performing a certain type of bridging this will be indicated by entries in the port table for the given type. |
| dot1dBasePort | .1.3.6.1.2.1.17 | The port number of the port for which this entry contains bridge management information. |
| dot1dPortIfIndex | .1.3.6.1.2.1.17 | The value of the instance of the <i>ifIndex</i> object, defined in <i>MIB-II</i> , for the interface corresponding to this port. |
| dot1dStpProtocolS pecification | .1.3.6.1.2.1.17 | An indication of what version of the Spanning Tree Protocol is being run. The value decLb100(2) indicates the DEC LANbridge 100 Spanning Tree protocol. IEEE 802.1d implementations will return ieee8021d(3). If future versions of the IEEE Spanning Tree Protocol are released that are incompatible with the current version a new value will be defined. |

| dot1dStpPriority | .1.3.6.1.2.1.17 | The value of the writeable portion of the <i>Bridge ID</i> , i.e., the first two octets of the (8 octet long) <i>Bridge ID</i> . The other (last) 6 octets of the <i>Bridge ID</i> are given by the value of <i>dot1dBaseBridgeAddress</i> . |
|----------------------------------|------------------------------|---|
| dot1dStpDesignate dRoot | .1.3.6.1.2.1.17 | The bridge identifier of the root of the spanning tree as determined by the <i>Spanning Tree Protocol</i> as executed by this node. This value is used as the <i>Root Identifier</i> parameter in all configuration <i>Bridge PDUs</i> originated by this node. |
| dot1dStpRootCost | .1.3.6.1.2.1.17 | The cost of the path to the root as seen from this bridge. |
| dot1dStpRootPort | .1.3.6.1.2.1.17 | The port number of the port which offers the lowest cost path from this bridge to the root bridge. |
| dot1dStpPort | .1.3.6.1.2.1.17 .2.15.1.1 | The port number of the port for which this entry contains Spanning Tree Protocol management information. |
| dot1dStpPortPriori ty | .1.3.6.1.2.1.17 .2.15.1.2 | The value of the priority field which is contained in the first (in network byte order) octet of the (2 octet long) Port ID. The other octet of the Port ID is given by the value of dot1dStpPort. |
| dot1dStpPortState | .1.3.6.1.2.1.17 .2.15.1.3 | The port's current state as defined by application of the <i>Spanning Tree Protocol</i> . This state controls what action a port takes on reception of a frame. If the bridge has detected a port that is malfunctioning it will place that port into the <i>broken(6)</i> state. For ports which are disabled (see <i>dot1dStpPortEnable</i>), this object will have a value of <i>disabled(1)</i> . |
| dot1dStpPortEnabl | .1.3.6.1.2.1.17 .2.15.1.4 | The enabled/disabled status of the port. |
| dot1dStpPortPathC ost | .1.3.6.1.2.1.17 .2.15.1.5 | The contribution of this port to the path cost of paths towards the spanning tree root which include this port. 802.1D-1990 recommends that the default value of this parameter be in inverse proportion to the speed of the attached LAN. |
| dot1dStpPortDesig natedRoot | .1.3.6.1.2.1.17 .2.15.1.6 | The unique <i>Bridge Identifier</i> of the <i>Bridge</i> recorded as the <i>Root</i> in the <i>Configuration BPDUs</i> transmitted by the <i>Designated Bridge</i> for the segment to which the port is attached. |
| dot1dStpPortDesig natedCost | .1.3.6.1.2.1.17 .2.15.1.7 | The path cost of the <i>Designated Port</i> of the segment connected to this port. This value is compared to the <i>Root Path Cost</i> field in received bridge <i>PDUs</i> . |
| dot1dStpPortDesig natedBridge | .1.3.6.1.2.1.17 .2.15.1.8 | The <i>Bridge Identifier</i> of the bridge which this port considers to be the <i>Designated Bridge</i> for this port's segment. |
| | | |

| dot1dStpPortDesig natedPort | .1.3.6.1.2.1.17 .2.15.1.9 | The <i>Port Identifier</i> of the port on the <i>Designated Bridge</i> for this port's segment. |
|--------------------------------|------------------------------|---|
| dot1dTpFdbAddres s | .1.3.6.1.2.1.17 | A unicast <i>MAC address</i> for which the bridge has forwarding and/or filtering information. |
| dot1dTpFdbPort | .1.3.6.1.2.1.17 | Either the value '0', or the port number of the port on which a frame having a source address equal to the value of the corresponding instance of <i>dot1dTpFdbAddress</i> has been seen. A value of '0' indicates that the port number has not been learned but that the bridge does have some forwarding/filtering information about this address (e.g. in the <i>dot1dStaticTable</i>). Implementors are encouraged to assign the port value to this object whenever it is learned even for addresses for which the corresponding value of <i>dot1dTpFdbStatus</i> is not <i>learned(3)</i> . |
| dot1dTpFdbStatus | .1.3.6.1.2.1.17 | The status of this entry. The meanings of the values are: other(1): none of the following. This would include the case where some other MIB object (not the corresponding instance of dot1dTpFdbPort, nor an entry in the dot1dStaticTable) is being used to determine if and how frames addressed to the value of the corresponding instance of dot1dTpFdbAddress are being forwarded. invalid(2): this entry is not longer valid (e.g., it was learned but has since aged-out), but has not yet been flushed from the table. learned(3): the value of the corresponding instance of dot1dTpFdbPort was learned, and is being used. self(4): the value of the corresponding instance of dot1dTpFdbAddress represents one of the bridge's addresses. The corresponding instance of dot1dTpFdbPort indicates which of the bridge's ports has this address. mgmt(5): the value of the corresponding instance of dot1dTpFdbAddress is also the value of an existing instance of dot1dStaticAddress. |

Table 96. Supported OIDS from the Q-BRIDGE-MIB

| Name | OID | Description |
|--------------------|--------------------|--|
| dot1qTpFdb Port | .1.3.6.1.2.1.17.7. | Either the value 0, or the port number of the port on which a frame having a source address equal to the value of the corresponding instance of dot1qTpFdbAddress has been seen. A value of 0 indicates that the port number has not been learned but that the device does have some forwarding/filtering information about this address (e.g., in the dot1qStaticUnicastTable). Implementors are encouraged to assign the port value to this object whenever it is learned, even for addresses for which the corresponding value of dot1qTpFdbStatus is not learned(3). |

.1.3.6.1.2.1.17.7. The status of this entry. dot1qTpFdb The 1.2.2.1.3 Status meanings of the values are: other(1): none of the following. This may include the case where some other MIB object (not the corresponding instance of dot1qTpFdbPort, nor an entry in the dot1qStaticUnicastTable) is being used to determine if and how frames addressed to the value of the corresponding instance of dot1qTpFdbAddress are being forwarded. invalid(2): this entry is no longer valid (e.g., it was learned but has since aged out), but has not yet been flushed from the table. *learned(3)*: the value of the corresponding instance of dot1qTpFdbPort was learned and is being used. *self(4)*: the value of the corresponding instance of dot1qTpFdbAddress represents one of the device's addresses. The corresponding instance of *dot1qTpFdbPort* indicates which of the device's ports has this address. *mgmt(5)*: the value of the corresponding instance of dot1qTpFdbAddress is also the value of an existing instance of dot1qStaticAddress.

Generic information about the *bridge* link discovery process can be found in the *Bridge Information* box on the *Node Detail Page* of the device. Information gathered from this *OID* will be stored in the following database table:

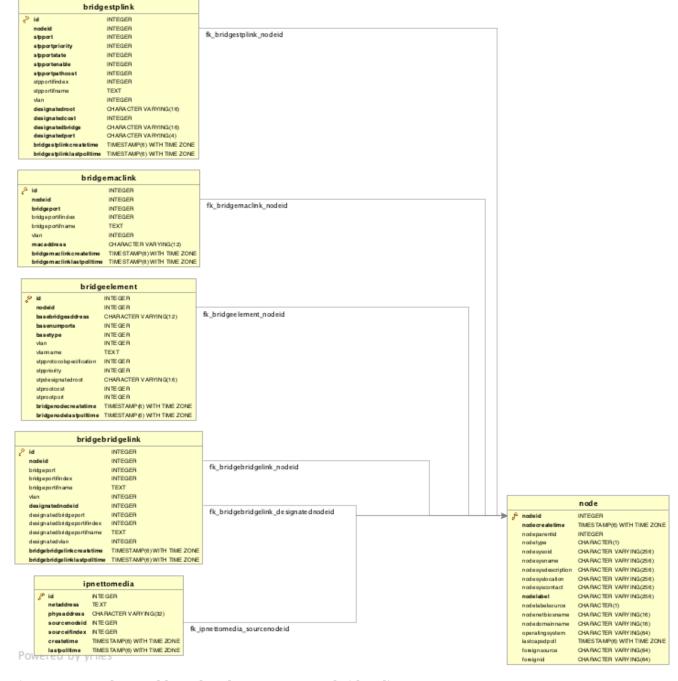


Figure 23. Database tables related to transparent bridge discovery

14.3. Layer 3 Link Discovery

With *Enlinkd* it is possible to get *Links* based on network routing applications. The following routing daemons can be used to provide a discovery of links based *Layer 3* information:

- Open Shortest Path First (OSPF)
- Intermediate System to Intermediate System (IS-IS)

This information is provided by *SNMP Agents* with appropriate *MIB support*. For this reason it is required to have a working *SNMP* configuration running. The link data discovered from *Enlinkd* is provided in the *Topology User Interface* and on the detail page of a node.

14.3.1. OSPF Discovery

The relevant MIBs for OSPF topology are *OSPF-MIB* and *OSPF-TRAP-MIB*. In these MIBs are defined the relevant objects used to find OSPF links, specifically:

- The Router ID which, in OSPF, has the same format as an IP address
- But identifies the router independent of its IP address.

Also all the interfaces are identified by their IP addresses. The OSPF links come from the SNMP ospfNbrTable defined in OSPF-MIB and this table is in practice persisted in the ospfLink table:

Table 97. Supported OIDs from OSPF-MIB

| Name | OID | Description |
|-----------------------------|----------------------------|--|
| ospfRouterId | .1.3.6.1.2.1.1 4.1.1.0 | A 32-bit integer uniquely identifying the router in the Autonomous System. By convention, to ensure uniqueness, this should default to the value of one of the router's IP interface addresses. This object is persistent and when written the entity should save the change to non-volatile storage. |
| ospfAdminStat | .1.3.6.1.2.1.1 4.1.2.0 | The administrative status of <i>OSPF</i> in the router. The value <i>enabled</i> denotes that the <i>OSPF Process</i> is active on at least one interface; <i>disabled</i> disables it on all interfaces. This object is persistent and when written the entity should save the change to non-volatile storage. |
| ospfVersionNum ber | .1.3.6.1.2.1.1 4.1.3.0 | The current version number of the OSPF protocol is 2. |
| ospfAreaBdrRtrS tatus | .1.3.6.1.2.1.1 | A flag to note whether this router is an Area Border Router. |
| ospfAreaASBdrR trStatus | .1.3.6.1.2.1.1 4.1.5.0 | A flag to note whether this router is configured as an Autonomous System Border Router. This object is persistent and when written the entity should save the change to non-volatile storage. |
| ospfIfIpAddress | .1.3.6.1.2.1.1 4.7.1.1 | The IP address of this OSPF interface. |
| ospfAddressLessI f | .1.3.6.1.2.1.1 4.7.1.2 | For the purpose of easing the instancing of addressed and addressless interfaces; this variable takes the value 0 on interfaces with IP addresses and the corresponding value of <i>ifIndex</i> for interfaces having no <i>IP address</i> . |
| ospfNbrIpAddr | .1.3.6.1.2.1.1 4.10.1.1 | The IP address this neighbor is using in its IP source address. Note that, on addressless links, this will not be 0.0.0.0 but the address of another of the neighbor's interfaces. |
| ospfNbrAddressL essIndex | .1.3.6.1.2.1.1 4.10.1.2 | On an interface having an <i>IP address</i> , zero. On addressless interfaces, the corresponding value of <i>ifIndex</i> in the <i>Internet Standard MIB</i> . On row creation, this can be derived from the instance. |

| Name | OID | Description |
|--------------|----------------------------|--|
| ospfNbrRtrId | .1.3.6.1.2.1.1 4.10.1.3 | A 32-bit integer (represented as a type <i>IpAddress</i>) uniquely identifying the neighboring router in the <i>Autonomous System</i> . |

Table 98. Supported OIDs from IP-MIB

| Name | OID | Description |
|--------------------|---------------------------|---|
| ipAdEntIfIn dex | .1.3.6.1.2.1.4. 20.1.2 | The index value which uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of the <i>IF-MIB</i> 's <i>ifIndex</i> . |
| ipAdEntNet Mask | .1.3.6.1.2.1.4. 20.1.3 | The subnet mask associated with the $IPv4$ address of this entry. The value of the mask is an $IPv4$ address with all the network bits set to 1 and all the hosts bits set to 0 . |

Generic information about the *OSPF* link discovery process can be found in the *OSPF Information* box on the *Node Detail Page* of the device. Information gathered from these OIDs will be stored in the following database table:



Figure 24. Database tables related to OSPF discovery

14.3.2. IS-IS Discovery

IS-IS Links are found in the isisISAdjTable that is defined in ISIS-MIB (mib-rfc4444.txt). In this table is found the information needed to find the Adjacency Intermediate System. The information about IS-IS is stored into two tables: isisElement and isisLink. isisElement contains the ISISSysID, a unique identifier of the "Intermediate System" (the name for the Router in ISO protocols). Each entry in this SNMP MIB table represents a unidirectional link from the Intermediate System that is queried to the Adjacent Intermediate Systems running IS-IS and "peering" with the source router. If two routers IS-A and IS-B support ISIS-MIB, then EnLinkd will create two link entries in OpenNMS Meridian: one from IS-A to IS-B (from the adjtable of IS-A) the complementary link back from IS-B to IS-A (from the

adjTable of _IS-B). IS-IS links are represented in the *ISIS-MIB* as follows:

Table 99. Supported OIDs from ISIS-MIB

| Name | OID | Description |
|--------------------------------|------------------|---|
| isisSysID | .1.3.6.1.2.1.138 | The ID for this Intermediate System. This value is appended to each of the area addresses to form the Network Entity Titles. The derivation of a value for this object is implementation specific. Some implementations may automatically assign values and not permit an SNMP write, while others may require the value to be set manually. Configured values must survive an agent reboot. |
| isisSysAdminStat e | .1.3.6.1.2.1.138 | The administrative state of this Intermediate System. Setting this object to the value on when its current value is off enables the Intermediate System. Configured values must survive an agent reboot. |
| isisSysObject | .1.3.6.1.2.1.138 | isisSysObject |
| isisCircIfIndex | .1.3.6.1.2.1.138 | The value of ifIndex for the interface to which this circuit corresponds. This object cannot be modified after creation. |
| isisCircAdminStat e | .1.3.6.1.2.1.138 | The administrative state of the circuit. |
| isisISAdjState | .1.3.6.1.2.1.138 | The state of the adjacency. |
| isisISAdjNeighSN PAAddress | .1.3.6.1.2.1.138 | The SNPA address of the neighboring system. |
| isisISAdjNeighSys Type | .1.3.6.1.2.1.138 | The type of the neighboring system. |
| isisISAdjNeighSys ID | .1.3.6.1.2.1.138 | The system ID of the neighboring Intermediate System. |
| isisISAdjNbrExte ndedCircID | .1.3.6.1.2.1.138 | The 4-byte <i>Extended Circuit ID</i> learned from the Neighbor during 3-way handshake, or 0. |

Generic information about the *IS-IS* link discovery process can be found in the *IS-IS* Information box on the *Node Detail Page* of the device. Information gathered from this OIDs will be stored in the following database table:

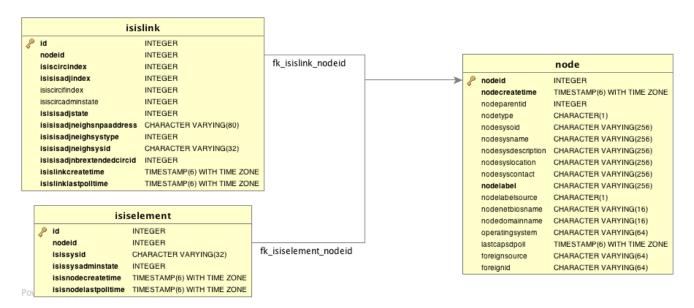


Figure 25. Database tables related to IS-IS discovery

Chapter 15. Operation

15.1. HTTPS / SSL

This chapter covers the possibilities to configure *OpenNMS Meridian* to protect web sessions with HTTPS and also explains how to configure *OpenNMS Meridian* to establish secure connections.



In order to use HTTPS the Java command line tool keytool is used. It is automatically shipped with each JRE installation. More details about the keytool can be found at the official documentation.

15.1.1. Standalone HTTPS with Jetty

To configure *OpenNMS Meridian* to protect web sessions with HTTPS please refer to the official *OpenNMS Meridian* Wiki article Standalone HTTPS with Jetty.

15.1.2. OpenNMS Meridian as HTTPS client

To establish secure HTTPS connections within Java one has to setup a so called Java Trust Store.

The *Java Trust Store* contains all certificates a Java application should trust when making connections as a client to a server.

Setup Java Trust Store

To setup the *Java Trust Store* the following command can be issued.



If you do not have a *Java Trust Store* setup yet, it is created automatically.

Import a certificate to the Java Trust Store

```
keytool \
  -import \ ①
  -v \ ②
  -trustcacerts \ ③
  -alias localhost \ ④
  -file localhost.cert \ ⑤
  -keystore /$OPENNMS_HOME/etc/trust-store.jks ⑥
```

- ① Define to import a certificate or a certificate chain
- ② Use verbose output
- 3 Define to trust certificates from cacerts
- 4 The alias for the certificate to import, e.g. the common name
- 5 The certificate to import
- 6 The location of the Java Trust Store

If you create a new *Java Trust Store* you are asked for a password to protect the *Java Trust Store*. If you update an already existing *Java Trust Store* please enter the password you chose when creating the *Java Trust Store* initially.

Download existing public certificate

To Download an existing public certificate the following command can be issued.

Download an existing public certificate

```
openssl \
   s_client \ ①
   -showcerts \ ②
   -connect localhost:443 \ ③
   -servername localhost \ ④
   < /dev/null \ ⑤
   > localhost.cert ⑥
```

- ① Use SSL/TLS client functionality of openss1.
- 2 Show all certificates in the chain
- 3 PORT:HOST to connect to, e.g. localhost:443
- 4 This is optional, but if you are serving multiple certificates under one single ip address you may define a server name, otherwise the ip of localhost:PORT certificate is returned which may not match the requested server name (mail.domain.com, opennms.domain.com, dns.domain.com)
- **5** No input
- 6 Where to store the certificate.

Configure OpenNMS Meridian to use the defined Java Trust Store

To setup *OpenNMS Meridian* to use the defined *Java Trust Store* the according <code>javax.net.ssl.trustStore*</code> properties have to be set. Open <code>\$OPENNMS_HOME/etc/opennms.properties</code> and add the properties <code>javax.net.ssl.trustStore</code> and <code>javax.net.ssl.trustStorePassword</code> as shown below.

\$OPENNMS_HOME/etc/opennms.properties snippet to define a Java Trust Store

```
javax.net.ssl.trustStore=/$OPENNMS_HOME/etc/trust-store.jks ①
javax.net.ssl.trustStorePassword=change-me ②
```

- 1 The location of the Java Trust Store
- 2 The password of the Java Trust Store

For more details on the Java build-in SSL System properties have a look at chapter Debugging / Properties.



Each time you modify the *Java Trust Store* you have to restart *OpenNMS Meridian* to have the changes take effect.

15.1.3. Differences between Java Trust Store and Java Key Store

The *Java Trust Store* is used to determine whether a remote connection should be trusted or not, e.g. whether a remote party is who it claims to be (client use case).

The *Java Key Store* is used to decide which authentication credentials should be sent to the remote host for authentication during SSL handshake (server use case).

For more details, please check the JSSE Reference Guide.

15.1.4. Debugging / Properties

If you encounter issues while using HTTPS it might be useful to enable debugging or use one of the build-in Java System Properties to configure the proper use of SSL.

Table 100. Java build-in System Properties (Source)

| System Property Name | Description |
|----------------------------------|---|
| javax.net.ssl.keyStore | Location of the Java keystore file containing an application process's own certificate and private key. On Windows, the specified pathname must use forward slashes, /, in place of backslashes, \. |
| javax.net.ssl.keyStorePassword | Password to access the private key from the keystore file specified by <code>javax.net.ssl.keyStore</code> . This password is used twice: to unlock the keystore file (store password) and to decrypt the private key stored in the keystore (key password). In other words, the JSSE framework requires these passwords to be identical. |
| javax.net.ssl.keyStoreType | (Optional) For Java keystore file format, this property has the value jks (or JKS). You do not normally specify this property, because its default value is already jks. |
| javax.net.ssl.trustStore | Location of the Java keystore file containing the collection of CA certificates trusted by this application process (trust store). On Windows, the specified pathname must use forward slashes, /, in place of backslashes, \. If a trust store location is not specified using this property, the Sun JSSE implementation searches for and uses a keystore file in the following locations (in order): \$JAVA_HOME/lib/security/jssecacerts and \$JAVA_HOME/lib/security/cacerts |
| javax.net.ssl.trustStorePassword | Password to unlock the keystore file (store password) specified by javax.net.ssl.trustStore. |

| System Property Name | Description |
|------------------------------|--|
| javax.net.ssl.trustStoreType | (Optional) For Java keystore file format, this property has the value jks (or JKS). You do not normally specify this property, because its default value is already jks. |
| javax.net.debug | To switch on logging for the SSL/TLS layer, set this property to ssl. More details about possible values can be found here. |

15.2. Geocoder Service

The *Geocoder Service* is used to resolve geolocation information within *OpenNMS Meridian*. By default The Google Map API is used to resolve the geolocation information, if available. In order to configure the Google Map API the following properties in etc/org.opennms.features.geocoder.google.cfg are supported:

| Proper ty | Туре | Default | Description | |
|---------------|-------------|-----------------|--|---------|
| clientI d | Strin g | empty string | The Google Map API Client ID. required if you exceed the free Google Map API usage. Please refer to the official documentation for more inform | This is |
| clientK ey | Strin g | empty string | The Google Map API API Key. This is required if you exceed the free Google Map API usage. Please refer to the official documentation for more information. | |
| timeout | Integ er | 500 | The connection timeout in milliseconds the Geocoder tries to resolve a single geolocation. | |

15.3. resourcecli: simple resource management tool

Sometimes a user want to list or manually delete collected data (resources) of an *OpenNMS Meridian* instance. When using *RRDTool*- or *JRobin*-based storage this can easily be achieved by traversing the share/rrd directory and its subdirectories. The several .rrd or .jrb files can be listed or deleted for individual nodes. When *Newts*-based storage is used the data is stored and indexed remotely on a *Cassandra* cluster. In this case the cluster must be queried for available resources. For the deletion of resources the data and all generated indexes must be gathered and removed. The *resourcecli* tool simplifies this process and works with *Newts*-based storage as well as with *RRDTool* and *JRobin* files.

15.3.1. Usage

The utility is installed by default and its wrapper script is located in the \${OPENNMS_HOME}/bin directory.

```
$ cd /path/to/opennms/bin
```

\$./resourcecli



When invoked without parameters the usage and help information is printed.

The *resourcecli* tool uses sub-commands for the different tasks. Each of these sub-commands provide different options and parameters. The command line tool accepts the following sub-commands.

| Sub- command | Description |
|-----------------|--|
| list | Queries an <i>OpenNMS Meridian</i> server for available resources. |
| show | Displays details for a given resource. |
| delete | Deletes a given resource and all of its child resources. |

The following global options are available in each of the sub-commands of the tool:

| Option/Argume nt | Description | Default |
|---------------------|---|-----------------------------------|
| help | Displays help and exit | false |
| username VALUE | Username for connecting to OpenNMS Meridian | admin |
| password VALUE | Password for connecting to OpenNMS Meridian | admin |
| url VALUE | URL of the <i>OpenNMS Meridian</i> instance to connect to | http://localhost:8980/openn ms |

15.3.2. Sub-command: list

This sub-command is used to query an *OpenNMS Meridian* instance for its available resources. The following example queries the local *OpenNMS Meridian* instance with the credentials admin/secret.

```
$ ./resourcecli --username admin --password secret list
node[72]
node[72].nodeSnmp[]
node[70].responseTime[192.168.0.2]
node[70].nodeSnmp[]
node[70].interfaceSnmp[bridge0]
node[70].interfaceSnmp[bridge1]
node[70].interfaceSnmp[vlan0-002500fe1bf3]
node[70].responseTime[50.16.15.18]
node[70].responseTime[192.168.0.1]
```

15.3.3. Sub-command: show

This sub-command can be used to show details for a given resource. The following example display details for the resource identified by resourceId node[70].

```
$ ./resourcecli --username admin --password secret show node\[70\]
ID:
            node[70]
            70
Name:
            MyRouter
Label:
            Node
Type:
Link:
            element/node.jsp?node=70
Parent ID:
            null
Children:
  node[70].nodeSnmp[]
  node[70].interfaceSnmp[bridge0]
  node[70].interfaceSnmp[bridge1]
  node[70].interfaceSnmp[vlan0-002500fe1bf3]
    node[70].responseTime[50.16.15.18]
  node[70].responseTime[192.168.0.1]
Attributes:
  External:
  Graphs:
  Strings:
```

The following options are available for the *show* sub-command.

| Option/Argume nt | Description | Defaul t |
|-----------------------|--|-------------|
| <resource></resource> | The resourceId of the resource to display. | - |

15.3.4. Sub-command: delete

This sub-command can be used to delete a given resource and its child resources. The following example deletes the resource identified by resourceId node[70]. When successful, this command does not generate any output.

```
$ ./resourcecli --username admin --password secret delete node\[70\]
$
```

The following options are available for the *delete* sub-command.

| Option/Argume nt | Description | Defaul t |
|-----------------------|---|-------------|
| <resource></resource> | The resourceId of the resource to be deleted. | - |

15.4. newts-repository-converter: Rrd/Jrb to Newts migration utility

This utility can be used to migrate existing *RRDTool*- or *JRobin*-based data to a *Newts* cluster. This will be achieved by traversing the share/rrd directory and its subdirectories, reading the data and properties files and persisting this data to *Newts*.

15.4.1. Migration

The following suggestions try to minimize the data collection gap that occur when reconfiguring *OpenNMS Meridian* for a different storage strategy. First, we determine the parameters needed for migration of the existing data. After that, we reconfigure *OpenNMS Meridian* to persists all new collected data to *Newts* storage. Finally, the *Rrd-* or *JRobin-*based data will be converted and persisted to *Newts* using the *newts-repository-converter* utility.

Prerequisites

- Working OpenNMS Meridian installation with *RRDTool* or *JRobin*-based storage strategy configured.
- Installed and working Newts cluster reachable by the OpenNMS Meridian instance.

Migration plan

- 1. Check and write down the values for the following options in your openms.properties file. You will need these information later to invoke the *newts-repository-converter* utility.
 - a. File etc/opennms.properties:
 - Check for the entry org.opennms.rrd.storeByGroup whether storeByGroup is enabled.
 - Check for the entry rrd.base.dir for the location where *Rrd* or *Jrb* files are stored.
 - Check for the entry rrd.binary for the location of the *RRDTool* binary.
 - b. File etc/rrd-configuration.properties:
 - Check for the entry org.opennms.rrd.strategyClass whether JRobinRrdStrategy (*JRobin*) or JniRrdStrategy / MultithreadedJniRrdStrategy (*RRDTool*) is used.
- 2. Stop your *OpenNMS Meridian* instance.
- 3. Reconfigure *OpenNMS Meridian* to persist data to *Newts* so, when correctly configured all new samples will be persisted into *Newts* after *OpenNMS Meridian* is started. Note, that the converter assumes storeByForeignSource to be enabled.
- 4. Start your *OpenNMS Meridian* instance.
- 5. Use the *newts-repository-converter* utility to convert the existing data to *Newts* by specifying the options that correspond to the information gathered during step #1.

This procedure will minimize the data collection gap to the time needed to reconfigure *OpenNMS Meridian* for *Newts* storage.



The *newts_converter* utility needs the path to the base directory of your *OpenNMS Meridian* instance for reading the configuration files. For instance the utility needs the datasource configuration during the migration process to query the database to lookup node data.

15.4.2. Usage

The utility is installed by default and its wrapper script is located in the \${OPENNMS_HOME}/bin directory.

- \$ cd /path/to/opennms/bin
- \$./newts-repository-converter



When invoked without parameters the usage and help information is printed.

The *newts-repository-converter* tool provide the following options and parameters:

| Short- option | Long- option | Description | Default |
|------------------|--------------------|---|----------------------------|
| h | help | Prints help and usage information | false |
| 0 | onms-home | OpenNMS Meridian Home Directory | /opt/opennms |
| Γ | rrd-dir | The path to the RRD data | ONMS- HOME/share/rrd |
| t | rrd-tool | Whether to use rrdtool or JRobin | |
| Т | rrd-binary | The binary path to the rrdtool command (only used if rrd-tool is set) | /usr/bin/rrdtool |
| S | store-by- group | Whether store by group was enabled or not | |
| n | threads | Number of conversion threads | defaults to number of CPUs |

15.4.3. Example 1: convert Rrd-based data with storeByGroup enabled

The following example shows how to convert *RRDTool*-based data that was stored with storeByGroup enabled. The OpenNMS Meridian home is <code>/opt/opennms</code>, the data directory is <code>/opt/opennms/share/rrd</code> and the *RRDTool* binary located at <code>/usr/local/bin/rrdtool</code>. This program call will use 16 concurrent threads to convert the *Rrd* files.

\$./newts-repository-converter -t true -s true -T /usr/local/bin/rrdtool -n 16
<output omitted>

15.4.4. Example 2: convert JRobin-based data with storeByGroup disabled

The following example shows how to convert *JRobin*-based data located in the directory /mnt/opennms/rrd that was collected with storeByGroup disabled. This program call will use 8 concurrent threads to convert the *Jrb* files.

\$./newts-repository-converter -t false -s false -r /mnt/opennms/rrd -n 8
<output omitted>

15.5. **Newts**

This section describes how to configure *OpenNMS Meridian* to use *Newts* and how to use *OpenNMS Meridian* to monitor your Cassandra cluster.

15.5.1. Configuration

Enabling Newts

OpenNMS Meridian can be configured to use Newts by setting the following property in in \${OPENNMS_HOME}/etc/opennms.properties:

```
org.opennms.timeseries.strategy=newts
```

It is also highly recommended that resources stored in *Newts* are referenced by their foreign source and foreign ID, as opposed to their database ID. To this end, the following property should also be set in the same file:

```
org.opennms.rrd.storeByForeignSource=true
```

With these set, *OpenNMS Meridian* will begin persisting metrics using the *Newts* engine when restarted.

Additional configuration options are presented in the next section.

Configuration Reference

The following properties, found in \${OPENNMS_HOME}/etc/opennms.properties, can be used to configure and tune *Newts*.

General

| Name | Default | Description |
|-----------------------------------|---------|------------------------------|
| org.opennms.newts.config.keyspace | newts | Name of the keyspace to use. |

| Name | Default | Description |
|--|-----------------|---|
| org.opennms.newts.config. hostname | localhost | IP address or hostnames of the Cassandra nodes. Multiple hosts can be separated by a comma. |
| org.opennms.newts.config. | 9042 | CQL port used to connect to the Cassandra nodes. |
| org.opennms.newts.config.username | cassandra | Username to use when connecting to Cassandra via CQL. |
| org.opennms.newts.config. password | cassandra | Password to use when connecting to Cassandra via CQL. |
| org.opennms.newts.config. | false | Enable/disable SSL when connecting to Cassandra. |
| org.opennms.newts.config.read_consistency | ONE | Consistency level used for <i>read</i> operations. See Configuring data consistency for a list of available options. |
| org.opennms.newts.config. write_consistency | ANY | Consistency level used for <i>write</i> operations. See Configuring data consistency for a list of available options. |
| org.opennms.newts.config. max_batch_size | 16 | Maximum number of records to insert in a single transaction. Limited by the size of the Cassandra cluster's batch_size_fail_threshold_in_kb property. |
| org.opennms.newts.config.ring_buffer_size | 8192 | Maximum number of records that can be held in the ring buffer. Must be a power of two. |
| org.opennms.newts.config. writer_threads | 16 | Number of threads used to pull samples from the ring buffer and insert them into Newts. |
| org.opennms.newts.config. | 31540000 | Number of seconds after which samples will automatically be deleted. Defaults to one year. |
| org.opennms.newts.config. resource_shard | 604800 | Duration in seconds for which samples will be stored at the same key. Defaults to 7 days in seconds. |
| org.opennms.newts.query.m inimum_step | 300000 | Minimum step size in milliseconds. Used to prevent large queries. |
| org.opennms.newts.query.i nterval_divider | 2 | If no interval is specified in the query, the step will be divided into this many intervals when aggregating values. |
| org.opennms.newts.query.h eartbeat | 450000 | Duration in milliseconds. Used when no heartbeat is specified. Should generally be 1.5x your largest collection interval. |
| org.opennms.newts.query.p arallelism | Number of cores | Maximum number of threads that can be used to compute aggregates. Defaults to the number of available cores. |
| org.opennms.newts.config. cache.strategy | See bellow | Canonical name of the class used for resource level caching. See the table bellow for all of the available options. |
| org.opennms.newts.config.cache.max_entries | 8192 | Maximum number of records to keep in the cache when using an in-memory caching strategy. |

| Name | Default | Description |
|---|---------|--|
| org.opennms.newts.nan_on_ counter_wrap | false | Disables the processing of counter wraps, replacing these with NaNs instead. |

Available caching strategies include:

| Name | Class | Defaul t |
|----------------------|--|-------------|
| In-Memory Cache | org.opennms.netmgt.newts.support.GuavaSearchableResourceMetadataCac he | Y |
| Redis-based Cache | org.opennms.netmgt.newts.support.RedisResourceMetadataCache | N |

Redis Cache

When enabled, the following options can be used to configure the Redis-based cache.

| Name | Default | Description |
|---|---------------|--|
| org.opennms.newts.config.cache.redis_hostna me | localhos t | IP address of hostname of the <i>Redis</i> server. |
| org.opennms.newts.config.cache.redis_port | 6379 | TCP port used to connect to the <i>Redis</i> server. |

Recommendations

You will likely want to change the values of cache.max_entries and the ring_buffer_size to suit your installation.

Meta-data related to resources are cached in order to avoid writing redundant records in *Cassandra*. If you are collecting data from a large number of resources, you should increase the cache.max_entries to reflect the number of resources you are collecting from, with a suitable buffer.

The samples gathered by the collectors are temporarily stored in a ring buffer before they are persisted to *Cassandra* using *Newts*. The value of the ring_buffer_size should be increased if you expect large peaks of collectors returning at once or latency in persisting these to *Cassandra*. However, note that the memory used by the ring buffer is reserved, and larger values may require an increased heap size.

15.5.2. Cassandra Monitoring

This section describes some of the metrics *OpenNMS Meridian* collects from a *Cassandra* cluster.



JMX must be enabled on the *Cassandra* nodes and made accessible from _*OpenNMS Meridian* in order to collect these metrics. See Enabling JMX authentication for details.



The data collection is bound to the agent IP interface with the service name *JMX-Cassandra*. The *JMXCollector* is used to retrieve the *MBean* entities from the *Cassandra* node.

Client Connections

The number of active client connections from org.apache.cassandra.metrics.Client are collected:

| Name | Description |
|------------------------|--------------------------------------|
| connectedNativeClients | Metrics for connected native clients |
| connectedThriftClients | Metrics for connected thrift clients |

Compaction Bytes

The following compaction manager metrics from org.apache.cassandra.metrics.Compaction are collected:

| Name | Description |
|----------------|--|
| BytesCompacted | Number of bytes compacted since node started |

Compaction Tasks

The following compaction manager metrics from org.apache.cassandra.metrics.Compaction are collected:

| Name | Description |
|----------------|--|
| CompletedTasks | Estimated number of completed compaction tasks |
| PendingTasks | Estimated number of pending compaction tasks |

Storage Load

The following storage load metrics from org.apache.cassandra.metrics.Storage are collected:

| Nam e | Description |
|----------|---|
| Load | Total disk space (in bytes) used by this node |

Storage Exceptions

The following storage exception metrics from org.apache.cassandra.metrics.Storage are collected:

| Name | Description |
|------------|--|
| Exceptions | Number of unhandled exceptions since start of this <i>Cassandra</i> instance |

Dropped Messages

Measurement of messages that were *DROPPABLE*. These ran after a given timeout set per message type so was thrown away. In *JMX* these are accessible via org.apache.cassandra.metrics.DroppedMessage. The number of dropped messages in the different message queues are good indicators whether a cluster can handle its load.

| Name | Stage | Description | |
|----------------------|--------------------------|--|--|
| Mutation | MutationStage | If a write message is processed after its timeout (write_request_timeout_in_ms) it either sent a failure to the client or it met its requested consistency level and will relay on hinted handoff and read repairs to do the mutation if it succeeded. | |
| Counter_Mut ation | MutationStage | If a write message is processed after its timeout (write_request_timeout_in_ms) it either sent a failure to the client or it met its requested consistency level and will relay on hinted handoff and read repairs to do the mutation if it succeeded. | |
| Read_Repair | MutationStage | Times out after write_request_timeout_in_ms. | |
| Read | ReadStage | Times out after read_request_timeout_in_ms. No point in servicing reads after that point since it would of returned error to client. | |
| Range_Slice | ReadStage | Times out after range_request_timeout_in_ms. | |
| Request_Res ponse | RequestRespons eStage | Times out after request_timeout_in_ms. Response was completed and sent back but not before the timeout | |

Thread pools

Apache Cassandra is based on a so called Staged Event Driven Architecture (SEDA). This seperates different operations in stages and these stages are loosely coupled using a messaging service. Each of these components use queues and thread pools to group and execute their tasks. The documentation for Cassandra Thread pool monitoring is originated from Pythian Guide to Cassandra Thread Pools.

Table 101. Collected metrics for Thread Pools

| Name | Description |
|-----------------------|--|
| ActiveTasks | Tasks that are currently running |
| CompletedTasks | Tasks that have been completed |
| CurrentlyBlockedTasks | Tasks that have been blocked due to a full queue |
| PendingTasks | Tasks queued for execution |

Memtable FlushWriter

Sort and write *memtables* to disk from org.apache.cassandra.metrics.ThreadPools. A vast majority of time this backing up is from over running disk capability. The sorting can cause issues as well however. In the case of sorting being a problem, it is usually accompanied with high load but a

small amount of actual flushes (seen in cfstats). Can be from huge rows with large column names, i.e. something inserting many large values into a *CQL* collection. If overrunning disk capabilities, it is recommended to add nodes or tune the configuration.



Alerts: pending > 15 | | blocked > 0

Memtable Post Flusher

Operations after flushing the *memtable*. Discard commit log files that have had all data in them in *sstables*. Flushing non-cf backed secondary indexes.



Alerts: pending > 15 | | blocked > 0

Anti Entropy Stage

Repairing consistency. Handle repair messages like merkle tree transfer (from Validation compaction) and streaming.



Alerts: pending > 15 | | blocked > 0

Gossip Stage

Post 2.0.3 there should no longer be issue with pending tasks. Instead monitor logs for a message:

Gossip stage has {} pending tasks; skipping status check ...

Before that change, in particular older versions of 1.2, with a lot of nodes (100+) while using vnodes can cause a lot of CPU intensive work that caused the stage to get behind. Been known to of been caused with out of sync schemas. Check *NTP* working correctly and attempt nodetool resetlocalschema or the more drastic deleting of system column family folder.



Alerts: pending > 15 | | blocked > 0

Migration Stage

Making schema changes



Alerts: pending > 15 | | blocked > 0

MiscStage

Snapshotting, replicating data after node remove completed.



Alerts: pending > 15 | | blocked > 0

Mutation Stage

Performing a local including:

- insert/updates
- · Schema merges
- · commit log replays
- · hints in progress

Similar to ReadStage, an increase in pending tasks here can be caused by disk issues, over loading a system, or poor tuning. If messages are backed up in this stage, you can add nodes, tune hardware and configuration, or update the data model and use case.



Alerts: pending > 15 | | blocked > 0

Read Stage

Performing a local read. Also includes deserializing data from row cache. If there are pending values this can cause increased read latency. This can spike due to disk problems, poor tuning, or over loading your cluster. In many cases (not disk failure) this is resolved by adding nodes or tuning the system.



Alerts: pending > 15 | | blocked > 0

Request Response Stage

When a response to a request is received this is the stage used to execute any callbacks that were created with the original request.



Alerts: pending > 15 | | blocked > 0

Read Repair Stage

Performing read repairs. Chance of them occurring is configurable per column family with read_repair_chance. More likely to back up if using CL.ONE (and to lesser possibly other non-CL.ALL queries) for reads and using multiple data centers. It will then be kicked off asynchronously outside of the queries feedback loop. Note that this is not very likely to be a problem since does not happen on all queries and is fast providing good connectivity between replicas. The repair being droppable also means that after write_request_timeout_in_ms it will be thrown away which further mitigates this. If pending grows attempt to lower the rate for high read CFs.



Alerts: pending > 15 | | blocked > 0

JVM Metrics

Some key metrics from the running Java virtual machine are also collected:

java.lang:type=Memory

The memory system of the Java virtual machine. This includes heap and non-heap memory

java.lang:type=GarbageCollector,name=ConcurrentMarkSweep

Metrics for the garbage collection process of the Java virtual machine



If you use *Apache Cassandra* for running *Newts* you can also enable additional metrics for the *Newts* keyspace.

15.5.3. Newts Monitoring

This section describes the metrics *OpenNMS Meridian* collects for monitoring the *Newts* keyspace from org.apache.cassandra.metrics.Keyspace on an *Cassandra* node.



JMX must be enabled on the *Cassandra* nodes and made accessible from _*OpenNMS Meridian* in order to collect these metrics. See Enabling JMX authentication for details.

The data collection is bound to the agent IP interface with the service name *JMX-Cassandra-Newts*. The *JMXCollector* is used to retrieve the *MBean* entities from the *Cassandra* node.

All Memory Table Data Size

| Name | Description |
|---------------------------------|--|
| AllMemtablesLiveData Size | Total amount of live data stored in the memtables (2i and pending flush memtables included) that resides off-heap, excluding any data structure overhead |
| AllMemtablesOffHeapD ataSize | Total amount of data stored in the memtables (2i and pending flush memtables included) that resides off-heap. |
| AllMemtablesOnHeapDa taSize | Total amount of data stored in the memtables (2i and pending flush memtables included) that resides on-heap. |

Memtable Switch Count

| Name | Description | |
|---------------------|--|--|
| MemtableSwitchCount | Number of times flush has resulted in the memtable being switched out. | |

Memtable Columns Count

| Name | Description |
|----------------------|--|
| MemtableColumnsCount | Total number of columns present in the memtable. |

Memory Table Data Size

| Name | Description |
|------|---|
| 70 | Total amount of live data stored in the memtable, excluding any data structure overhead |

| Name | Description |
|-----------------------------|--|
| MemtableOffHeapDat aSize | Total amount of data stored in the memtable that resides off-heap, including column related overhead and partitions overwritten. |
| MemtableOnHeapData Size | Total amount of data stored in the memtable that resides on-heap, including column related overhead and partitions overwritten. |

Read and Write Latency

| Name | Description |
|-------------------|----------------------|
| ReadTotalLatency | Local read metrics. |
| WriteTotalLatency | Local write metrics. |

Range Latency

| Name | Description |
|------------------------------|--|
| RangeLatency 99th Percentile | Local range slice metrics 99th percentile. |

Latency

| Name | Descriptio n |
|------------------------|-----------------|
| CasCommitTotalLatency | |
| CasPrepareTotalLatency | |
| CasProposeTotalLatency | |

Bloom Filter Disk Space

| Name | Description |
|--------------------------|---------------------------------|
| BloomFilterDiskSpaceUsed | Disk space used by bloom filter |

Bloom Filter Off Heap Memory

| Name | Description |
|------------------------------|--------------------------------------|
| BloomFilterOffHeapMemoryUsed | Off heap memory used by bloom filter |

Newts Memory Used

| Name | Description |
|--------------------------------------|---|
| CompressionMetadataOffHeapMemoryUsed | Off heap memory used by compression meta data |
| IndexSummaryOffHeapMemoryUsed | Off heap memory used by index summary |

Pending

| Name | Description | |
|--------------------|--|--|
| PendingCompactions | Estimate of number of pending compactions for this column family | |
| PendingFlushes | Estimated number of tasks pending for this column family | |

Disk Space

| Name | Description |
|------------------------|---|
| TotalDiskSpaceU sed | Total disk space used by <i>SSTables</i> belonging to this column family including obsolete ones waiting to be garbage collected. |
| LiveDiskSpaceUs ed | Disk space used by SSTables belonging to this column family |

15.6. Daemon Configuration Files

Configuration changes require a restart of OpenNMS and some daemons are able to reload configuration changes triggered by a daemon reload event. This section gives an overview about all daemons and the related configuration files and which can be reloaded without restarting OpenNMS.

15.6.1. Eventd

| Internal Daemon Name | Reload Event |
|-------------------------|--|
| Eventd | uei.opennms.org/internal/reloadDaemonConfig -p 'daemonName Eventd' |

Table 102. Eventd configuration file overview

| File | Restart Required | Reload Event | Description |
|----------------------------------|---------------------|-----------------|---|
| eventd- configuration.x ml | yes | no | Configure generic behavior of <i>Eventd</i> , i.e. <i>TCP</i> and <i>UDP</i> port numbers with <i>IP addresses</i> to listen for <i>Events</i> and socket timeouts. |
| eventconf.xml | no | yes | Main configuration file for <i>Eventd</i> . |
| events/* | no | yes | Out-of-the-box, all files in this folder are included via include directives in eventconf.xml. |

15.6.2. Notifd

| Internal Daemon Name | Reload Event |
|-------------------------|--|
| Notifd | uei.opennms.org/internal/reloadDaemonConfig -p 'daemonName Notifd' |

 ${\it Table~103.~Notifd~configuration~file~overview}$

| File | Restart Required | Reload Event | Description |
|---|---------------------|-----------------|---|
| notifd- configuration.xml | no | yes | Describes auto-acknowledge prefix, e.g. prefix "RESOLVED: " for nodeUp/nodeDown events. |
| notificationCommands .xml | no | no | Configuration for notification media, e.g. scripts, XMPP or HTTP Post, immediately applied. |
| notifications.xml | no | no | Event notification definitions and changes are immediately applied. |
| destinationPaths.xml | no | no | Contains paths for notification targets, e.g. JavaMail, XMPP or external scripts. |
| users.xml | no | no | Contain pager and address information for notification destination paths. |
| groups.xml | no | no | Groups can be used as target for notifications. |
| javamail- configuration.proper ties | no | no | Configuration to send notification mails via specific mail servers. |

15.6.3. Pollerd

| Internal Daemon Name | Reload Event |
|-------------------------|---|
| Pollerd | uei.opennms.org/internal/reloadDaemonConfig -p 'daemonName Pollerd' |

Table 104. Pollerd configuration file overview

| File | Restart Required | Reload Event | Description |
|-----------------------------------|---------------------|-----------------|--|
| poller- configuration.x ml | yes | yes | Restart is required in case new monitors are created or removed. Reload Event loads changed configuration parameters of existing monitors. |
| response- graph.propertie s | no | no | Graph definition for response time graphs from monitors |
| <pre>poll- outages.xml</pre> | no | yes | Can be reloaded with uei.opennms.org/internal/schedOutagesChanged |

Chapter 16. System Properties

The global behavior of *OpenNMS* is configured with Property files. Configuration can have also effect on the *Java Virtual Machine* underneath *OpenNMS*. Changes in these property files require a restart of *OpenNMS*. The configuration files can be found in \${OPENNMS_HOME}/etc.

The priority for *Java system properties* is as follows:

- 1. Those set via the Java command line i.e. in opennms.conf via ADDITIONAL_MANAGER_OPTIONS
- 2. opennms.properties.d/*.properties
- 3. opennms.properties
- 4. libraries.properties
- 5. rrd-configuration.properties
- 6. bootstrap.properties

Property files in opennms.properties.d/ are sorted alphabetically.



To avoid conflicts with customized configurations, all custom properties can be added to one or more files in \${OPENNMS_HOME}/etc/opennms.properties.d/. It is recommended to avoid modification of OpenNMS properties from the default installation. Create dedicated files with your customized properties in opennms.properties.d/.

Chapter 17. Ticketing

The ticketing integration allows *OpenNMS Meridian* to create trouble tickets in external systems. Tickets can be created and updated in response to new and/or resolved alarms.

To activate the ticketing integration, the following properties in \${OPENNMS_HOME}/etc/opennms.properties must be set accordingly:

| Property | Default | Description |
|--|------------------------|--|
| opennms.ticketer.plugin | NullTickete rPlugin | The plugin implementation to use. Each ticketer integration should define which value to set. The NullTicketerPlugin does nothing when attempting to create/update/delete tickets. |
| opennms.alarmTroubleTick etEnabled | false | Defines if the integration is enabled. If enabled various links to control the issue state is shown on the alarm details page. |
| opennms.alarmTroubleTi cketLinkTemplate | \${id} | A template to generate a link to the issue, e.g. \${id} |

17.1. JIRA Ticketing Plugin

The JIRA Ticketing Plugin is used to create JIRA Issues in response to OpenNMS Meridian alarms.

17.1.1. Setup

First, you'll need to install the opennms-plugin-ticketer-jira package for your system. The JIRA ticketing plugin and it's dependencies are not part of the core packages.

Now, in order to enable the plugin start by setting following property in \${OPENNMS_HOME}/etc/opennms.properties:

```
opennms.ticketer.plugin=org.opennms.netmgt.ticketd.OSGiBasedTicketerPlugin
```

Configure the plugin options by setting the following properties in \${OPENNMS_HOME}/etc/jira.properties:

| Name | Description |
|---------------|---|
| jira.host | JIRA Server Url |
| jira.username | Username |
| jira.password | Password |
| jira.project | The key of the project to use. Use jira:list-projects command to determine the project key. |

| Name | Description |
|---------------------------|---|
| jira.type | The Issue Type Id to use when opening new issues. Use jira:list-issue-types command to determine the issue type id. |
| jira.resolve | Name of the transition to use when resolving issues |
| jira.reopen | Name of the transition to use when re-opening issues |
| jira.status.open | Comma-separated list of JIRA status names for which the ticket should be considered 'Open' |
| jira.status.close d | Comma-separated list of JIRA status names for which the ticket should be considered 'Closed' |
| jira.status.cance lled | Comma-separated list of JIRA status names for which the ticket should be considered 'Cancelled' |
| jira.cache.reload Time | The time in milliseconds it takes to reload the <i>fields cache</i> . This is required to prevent the plugin to read the issue type's meta data every time an issue is created. A value of 0 disables the cache. Default value is 300000 (5 minutes). |



The transition names for resolve and reopen are typically found on buttons when looking at the ticket in JIRA



Either use jira:list-issue-types *OSGI Command* or https://confluence.atlassian.com/display/JIRA050/Finding+the+Id+for+Issue+Types for determining the appropriate issue type id.

Next, add jira-troubleticketer to the featuresBoot property in the \${OPENNMS_HOME}/etc/org.apache.karaf.features.cfg

Restart OpenNMS Meridian.

When *OpenNMS Meridian* has started again, login to the *Karaf Shell* and install the feature:

```
features:install jira-troubleticketer
```

The plugin should be ready to use.

17.1.2. Jira Commands

The JIRA Ticketing Plugin provides various OSGI Commands which can be used on the Karaf Shell to help set up the plugin.

There are OSGI Commands to list all available projects, versions, components, groups, issue types and even more.

To list all available commands simply type help | grep jira in the *Karaf Shell*.

Afterwards you can type for example jira:list-projects --help to determine the usage of a

command.

17.1.3. Custom fields

The *OpenNMS Meridian* Ticketer model is limited to the most common fields provided by all ticketing systems.

Besides the common fields creator, create date, description or subject, ticket system proprietary fields usually need to be set.

In some cases, even additional - so called - custom fields are defined.

In order to set these fields, the *JIRA Ticketing Plugin* provides the possibility to define those in the OpenNMS Ticket attributes which can be overwritten with the Usage of Drools.

To enable the Drools Ticketing integration, the following property in \${OPENNMS_HOME}/etc/opennms.properties must be set:

```
opennms.ticketer.servicelayer=org.opennms.netmgt.ticketd.DroolsTicketerServiceLayer
```

In addition the property in \${OPENNMS_HOME/etc/drools-ticketer.properties must point to a drools-ticketer-rules.drl file:

```
drools-ticketer.rules-file=${OPENNMS_HOME/etc/drools-ticketer-rules.drl
```

Finally a Drools Rule file named drools-ticketer-rules.drl must be placed in \${OPENNMS_HOME}/etc.

The following drools example snippet defines attributes to set custom fields:

Fields must be referenced by their id. To identify the id of a field, the jira:list-fields command can be used. By default only custom fields are shown. The -s options allows to show all fields. This may be necessary if JIRA default values need to be set as well, e.g. the Component, the Reporter, the Asignee, etc. Even the project key or issue type can be defined differently than originally in the jira.properties.

The *OpenNMS Ticketer Attribute* model only allows to set a String value. However the JIRA model is slightly different. Therefore each String value must be converted to a JIRA field type. The following table describes valid values for an OpenNMS attribute.

| Туре | Description |
|-----------------------|---|
| any | Any string. |
| date | Any date in the format of YYYY-MM-DD. |
| datetime | Any datetime in ISO 8601 format: YYYY-MM-DDThh:mm:ss.sTZD. |
| group | The name of the group. |
| user | The name of the user. |
| project | The key of the project (e.g. NMS) |
| version | The name of the version. To list all available versions, use jira:list-versions. |
| string | Any string. |
| option | The name of the option. |
| issuetype | The name of the issuetpye, e.g. Bug. To list all issue types, use jira:list-issuetypes. |
| priority | The name of the priority, e.g. Major. To list all priorites, use jira:list-priorities. |
| option-with- child | Either the name of the option, or a comma separated list (e.g. parent, child). |
| number | Any valid number (e.g. 1000) |
| array | If the type is array the value must be of the containing type. E.g. to set a custom field which defines multiple groups, the value jira-users, jira-administrators is mapped properly. The same is valid for versions: 18.0.3,19.0.0. |

As described above the values are usually identified by their name instead of their id (projects are identified by their key). This is easier to read, but may break the mapping code, if for example the name of a component changes in the future. To change the mapping from name (or key) to id an entry in jira.properties must be made:

```
jira.attributes.customfield_10113.resolution=id
```

To learn more about the Jira REST API please consult the following pages:

- https://developer.atlassian.com/jiradev/jira-apis/jira-rest-apis/jira-rest-api-tutorials/jira-rest-api-example-create-issue#JIRARESTAPIExample-CreateIssue-MultiSelect
- https://docs.atlassian.com/jira/REST/cloud/

The following jira (custom) fields have been tested with jira version 6.3.15:

Checkboxes

- Date Picker
- Date Time Picker
- Group Picker (multiple groups)
- Group Picker (single group)
- Labels
- · Number Field
- Project Picker (single project)
- Radio Buttons
- Select List (cascading)
- Select List (multiple choices)
- Select List (single choice)
- Text Field (multi-line)
- Text Field (read only)
- Text Field (single line)
- URL Field
- User Picker (multiple user)
- User Picker (single user)
- Version Picker (multiple versions)
- Version Picker (single version)



All other field types are mapped as is and therefore may not work.

Examples

The following output is the result of the command jira:list-fields -h http://localhost:8080 -u admin -p testtest -k DUM -i Bug -s and lists all available fields for project with key DUM and issue type Bug:

| Name | Id | Custom | Туре |
|-------------------|-------------------|--------|-------------|
| Affects Version/s | versions | false | array |
| Assignee | assignee | false | user |
| Attachment | attachment | false | array |
| Component/s | components | false | array 🕦 |
| Description | description | false | string |
| Environment | environment | false | string |
| Epic Link | customfield_10002 | true | any |
| Fix Version/s | fixVersions | false | array ② |
| Issue Type | issuetype | false | issuetype ③ |
| Labels | labels | false | array |
| Linked Issues | issuelinks | false | array |
| Priority | priority | false | priority ④ |
| Project | project | false | project (5) |
| Reporter | reporter | false | user |
| Sprint | customfield_10001 | true | array |
| Summary | summary | false | string |
| custom checkbox | customfield_10100 | true | array ⑥ |
| custom datepicker | customfield_10101 | true | date |

- 1 Defined Components are core, service, web
- 2 Defined versions are 1.0.0 and 1.0.1
- 3 Defined issue types are Bug and Task
- 4 Defined priorities are Major and Minor
- 5 Defined projects are NMS and HZN
- 6 Defined options are yes, no and sometimes

The following snipped shows how to set the various custom fields:

```
ticket.addAttribute("components", "core,web"); 1
ticket.addAttribute("assignee", "ulf"); 2
ticket.addAttribute("fixVersions", "1.0.1"); 3
ticket.addAttribte("issueType", "Task"); 4
ticket.addAttribute("priority", "Minor"); 5
ticket.addAttribute("project", "HZN"); 6
ticket.addAttribute("summary", "Custom Summary"); 7
ticket.addAttribute("customfield_10100", "yes,no"); 8
ticket.addAttribute("customfield_10101", "2016-12-06"); 9
```

- ① Sets the components of the created issue to core and web.
- ② Sets the Asignee of the issue to the user with login ulf.
- 3 Sets the fix version of the issue to 1.0.1
- 4 Sets the issue type to Task, overwriting the value of jira.type.
- (5) Sets the priority of the created issue to Minor.
- 6 Sets the project to HZN, overwriting the value of jira.project.

- The Sets the summary to Custom Summary, overwriting any previous summary.
- 8 Checks the checkboxes yes and no.
- Sets the value to 2016-12-06.

17.1.4. Troubleshooting

When troubleshooting, consult the following log files:

- \${OPENNMS_HOME}/data/log/karaf.log
- \${OPENNMS_HOME}/logs/trouble-ticketer.log

You can also try the jira: verify OSGI Command to help identifying problems in your configuration.

17.2. Remedy Ticketing Plugin

The *Remedy Ticketing Plugin* is used to create requests in the BMC Remedy ARS Help Desk Module in response to *OpenNMS Meridian* alarms.

17.2.1. Remedy Product Overview

It's important to be specific when discussing Remedy, because BMC Remedy is a suite of products. The *OpenNMS Meridian* Remedy Ticketing Plugin requires the core Remedy ARS and the Help Desk Module. The Help Desk Module contains a Help Desk Interface Web Service, which serves as the endpoint for creating, updating, and fetching tickets.

The Help Desk Interface (HDI) Web Service requires extensive configuration for its basic operation, and may need additional customization to interoperate with the *OpenNMS Meridian* Remedy Ticketing Plugin. Contact your Remedy administrator for help with required configuration tasks.

17.2.2. Supported Remedy Product Versions

Currently supported Remedy product versions are listed below:

| Product | Version |
|------------------|--------------------------|
| Remedy ARS | 7.6.04 Service Pack 2 |
| Help Desk Module | 7.6.04 Service Pack 1 |
| HDI Web Service | Same as Help Desk Module |

17.2.3. Setup

The Remedy Ticketing Plugin and its dependencies are part of the *OpenNMS Meridian* core packages.

Start by enabling the plugin and the ticket controls in the *OpenNMS Meridian* web interface, by setting the following properties in \${OPENNMS_HOME}/etc/opennms.properties:

opennms.ticketer.plugin=org.opennms.netmgt.ticketer.remedy.RemedyTicketerPlugin opennms.alarmTroubleTicketEnabled = true

In the same file, set the property opennms.alarmTroubleTicketLinkTemplate to a value appropriate for constructing a link to tickets in the Remedy web interface. A sample value is provided but must be customized for your site; the token \${id} will be replaced with the Remedy ticket ID when the link is rendered.

Now configure the plugin itself by setting the following properties in \${OPENNMS_HOME}/etc/remedy.properties:

| Name | Requi red | Description |
|--|--------------|--|
| remedy.username | requir ed | Username for authenticating to Remedy |
| remedy.password | requir ed | Password for authenticating to Remedy |
| remedy.authentication | option al | Authentication style to use |
| remedy.locale | option al | Locale for text when creating and updating tickets |
| remedy.timezone | option al | Timezone for interaction with Remedy |
| remedy.endpoint | requir ed | The endpoint URL of the HPD web service |
| remedy.portname | requir ed | The Port name of the HPD web service |
| remedy.createendpoint | requir ed | The endpoint location of the Create-HPD web service |
| remedy.createportname | requir ed | The Port name of the Create-HPD web service |
| remedy.targetgroups | option al | Colon-separated list of Remedy groups to which created tickets may be assigned ({group} below refers to values from this list) |
| <pre>remedy.assignedgroup.{grou p}</pre> | option al | Assigned group for the target group {group} |
| <pre>remedy.assignedsupportcomp any.{group}</pre> | option al | Assigned support company for the target group {group} |
| <pre>remedy.assignedsupportorga nization.{group}</pre> | option al | Assigned support organization for the target group {group} |

| Name | Requi red | Description |
|--|--------------|--|
| remedy.assignedgroup | requir ed | Default group to assign the ticket in case the ticket itself lacks information about a target assigned group |
| remedy.firstname | requir ed | First name for ticket creation and updating. Must exist in Remedy. |
| remedy.lastname | requir ed | Last name for ticket creation and updating. Must exist in Remedy. |
| remedy.serviceCI | requir ed | A valid Remedy Service CI for ticket creation |
| remedy.serviceCIReconID | requir ed | A valid Remedy Service CI Reconciliation ID for ticket creation |
| remedy.assignedsupportcomp any | requir ed | A valid default assigned support company for ticket creation |
| remedy.assignedsupportorga nization | requir ed | A valid default assigned support organization for ticket creation |
| remedy.categorizationtier1 | requir ed | A valid categorization tier (primary) for ticket creation |
| remedy.categorizationtier2 | requir ed | A valid categorization tier (secondary) for ticket creation |
| remedy.categorizationtier3 | requir ed | A valid categorization tier (tertiary) for ticket creation |
| remedy.serviceType | requir ed | A valid service type for ticket creation |
| remedy.reportedSource | requir ed | A valid Reported Source for ticket creation |
| remedy.impact | requir ed | A valid value for Impact, used in ticket creation |
| remedy.urgency | requir ed | A valid value for Urgency, used in ticket creation |
| remedy.reason.reopen | requir ed | The reason code set in Remedy when the ticket is reopened in <i>OpenNMS Meridian</i> |
| remedy.resolution | requir ed | The reason code set in Remedy when the ticket is closed in OpenNMS Meridian |
| remedy.reason.cancelled | requir ed | The reason code set in Remedy when the ticket is cancelled in <i>OpenNMS Meridian</i> |



The values for many of the required properties are site-specific; contact your Remedy administrator for assistance.

Restart OpenNMS Meridian.

The plugin should be ready to use. When troubleshooting, consult the following log files:

• \${OPENNMS_HOME}/logs/trouble-ticketer.log

17.3. TSRM Ticketing Plugin

The TSRM Ticketing Plugin is used to create TSRM incidents in response to OpenNMS Meridian alarms.

17.3.1. Setup

In order to enable the plugin start by setting following property in \${OPENNMS_HOME}/etc/opennms.properties:

```
opennms.ticketer.plugin=org.opennms.netmgt.ticketd.OSGiBasedTicketerPlugin
```

Configure the plugin options by setting the following properties in \${OPENNMS HOME}/etc/tsrm.properties:

| Name | Description |
|-------------------|-------------------------------|
| tsrm.url | TSRM Endpoint URL |
| tsrm.ssl.strict | Strict SSL Check (true/false) |
| tsrm.status.open | TSRM status for open ticket |
| tsrm.status.close | TSRM status for close ticket |

Next, add tsrm-troubleticketer to the featuresBoot property in the \${OPENNMS_HOME}/etc/org.apache.karaf.features.cfg

Restart OpenNMS.

When *OpenNMS* has started again, login to the *Karaf Shell* and install the feature:

```
features:install tsrm-troubleticketer
```

The plugin should be ready to use. When troubleshooting, consult the following log files:

- \${OPENNMS HOME}/data/log/karaf.log
- \${OPENNMS HOME}/logs/trouble-ticketer.log

17.3.2. Mapping OpenNMS Ticket with TSRM Incident

Following tables shows mapping between OpenNMS ticket and TSRM Incident

| Ticket Field | TSRM Incident Field |
|-----------------|----------------------------|
| id | TICKETID |
| state | STATUS |
| summary | DESCRIPTION |
| details | DESCRIPTIONLONGDESCRIPTION |
| user | REPORTEDBY |

Below fields are not part of Ticket, they have to be added as attributes.

| Ticket Field | TSRM Incident Field |
|--------------------|------------------------|
| affectedPerson | AFFECTEDPERSON |
| assetNum | ASSETNUM |
| classId | CLASS |
| classStructureId | CLASSSTRUCTUREID |
| commodity | COMMODITY |
| location | LOCATION |
| ownerGroup | OWNERGROUP |
| shsCallerType | SHSCALLERTYPE |
| shsReasonForOutage | SHSREASONFOROUTAGE |
| shsResolution | SHSRESOLUTION |
| shsRoomNumber | SHSROOMNUMBER |
| siteId | SITEID |
| source | source |
| statusIface | STATUSIFACE |

Chapter 18. Enabling RMI

By default, the RMI port in the OpenNMS Meridian server is disabled, for security reasons. If you wish to enable it so you can access OpenNMS Meridian through jconsole, remote-manage OpenNMS Meridian, or use the remote poller over RMI, you will have to add some settings to the default OpenNMS Meridian install.

18.1. Enabling RMI

To enable the RMI port in OpenNMS Meridian, you will have to add the following to the \${OPENNMS HOME}/etc/opennms.conf file. If you do not have an opennms.conf file, you can create it.

```
# Configure remote JMX
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dcom.sun.management.jmxremote.port=18980"
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dcom.sun.management.jmxremote.local.only=false"
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dcom.sun.management.jmxremote.authenticate=true"
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dcom.sun.management.jmxremote.ssl=false"

# Listen on all interfaces
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dopennms.poller.server.serverHost=0.0.0.0"
# Accept remote RMI connections on this interface
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Djava.rmi.server.hostname=<your-server-ip-address>"
```

This tells OpenNMS Meridian to listen for RMI on port 18980, and to listen on all interfaces. (Originally, RMI was only used for the Remote Poller, so despite the porperty name mentioning the "opennms poller server" it applies to RMI as a whole.) Note that you *must* include the -Djava.rmi.server.hostname= option or OpenNMS Meridian will accept connections on the RMI port, but not be able to complete a valid connection.

Authentication will only be allowed for users that are in the admin role (i.e. ROLE_ADMIN), or the jmx role (i.e. ROLE_JMX). To make a user an admin, be sure to add only the ROLE_ADMIN role to the user in users.xml. To add the jmx role to the user, add the ROLE_JMX role to the user in users.xml, and also the ROLE_USER role if is required to provide access to the WebUI.

Make sure \$OPENNMS_HOME/etc/jmxremote.access has the appropriate settings:

```
admin readwrite
jmx readonly
```

The possible types of access are:

readwrite

Allows retrieving JMX metrics as well as executing MBeans.

readonly

Allows retrieving JMX metrics but does **not** allow executing MBeans, even if they just return simple values.

18.2. Enabling SSL

To enable SSL on the RMI port, you will need to have an existing keystore for the OpenNMS Meridian server. For information on configuring a keystore, please refer to the official *OpenNMS Meridian* Wiki article Standalone HTTPS with Jetty.

You will need to change the com.sun.management.jmxremote.ssl option to true, and tell OpenNMS Meridian where your keystore is.

```
# Configure remote JMX
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dcom.sun.management.jmxremote.port=18980"
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dcom.sun.management.jmxremote.local.only=false"
ADDITIONAL MANAGER OPTIONS="$ADDITIONAL MANAGER OPTIONS
-Dcom.sun.management.jmxremote.authenticate=true"
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dcom.sun.management.jmxremote.ssl=true"
# Configure SSL Keystore
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Djavax.net.ssl.keyStore=/opt/opennms/etc/opennms.keystore"
ADDITIONAL MANAGER OPTIONS="$ADDITIONAL MANAGER OPTIONS
-Djavax.net.ssl.keyStorePassword=changeit"
# Listen on all interfaces
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Dopennms.poller.server.serverHost=0.0.0.0"
# Accept remote RMI connections on this interface
ADDITIONAL_MANAGER_OPTIONS="$ADDITIONAL_MANAGER_OPTIONS
-Djava.rmi.server.hostname=<your-server-ip-address>"
```

18.3. Connecting to RMI over SSL

Note that if you are using a self-signed or otherwise untrusted certificate, you will need to configure a *truststore* on the client side when you attempt to connect over SSL-enabled RMI. To create a truststore, follow the example in the HTTPS client instructions in the operator section of the manual. You may then use the truststore to connect to your OpenNMS Meridian RMI server.

For example, when using jconsole to connect to the OpenNMS Meridian RMI interface to get JVM statistics, you would run:

jconsole -J-Djavax.net.ssl.trustStore=/path/to/opennms.truststore -J
-Djavax.net.ssl.trustStorePassword=changeit

Chapter 19. Plugin Manager

With the introduction of *Karaf* as an *OSGi* application container, *OpenNMS Meridian* now has the ability to install or upgrade features on top of a running instance of *OpenNMS Meridian*. In addition, the new distributed *OSGi* architecture allows an *OpenNMS Meridian* system to be deployed as multiple software modules each running in their own *Karaf* instance.

The *OpenNMS Meridian* Plugin Manager_ provides a unified interface for managing the lifecycle of optional *OSGi* plugins installed in *OpenNMS Meridian* or in any *Karaf* instances which it manages. This need not be limited to *Karaf* instances running *OpenNMS Meridian* but can also be used to deploy modules to *Karaf* instances running user applications.

In addition to managing the installation of *OSGi* features, the *Plugin Manager* also allows the installation of licence keys which can be used to enable features for a particular instance of *OpenNMS Meridian*. Although the *OpenNMS Meridian* platform remains open source, this provides a mechanism for third parties developing features on top of the *OpenNMS Meridian* platform to manage access to their software.

The *Plugin Manager* also provides a mechanism for a separate 'app-store' or Available Plugins Server to be used to deliver these new features and / or licences into a particular *OpenNMS Meridian* instance. It is also possible to deliver software without access to the internet using the traditional *Karaf* Kar/RPM deployment model. (Kar files are a form of zip file containing bundles and features definitions which can be deployed in the *Karaf* /deploy directory). These can be placed in the /deploy directory directly or installed there using an RPM). In this case a number of features can be delivered together in a single software package but each only enabled at run time using the Plugin Manager.

OpenNMS Meridian plugins are standard *Karaf* features with additional metadata which describes the feature and the licence (if any) required. A plugin requiring a licence will not start if a valid licence string is not also installed.

Note that *Karaf*'s features mechanism has not been modified in any way. The Plugin Manager simply provides a user front end and additional metadata for features. Plugin features can be installed from the internal features repository, remote maven repositories or *Kar* files placed in the deploy directory depending on how the *Karaf* configuration is set up. The standard *OpenNMS Meridian* configuration has no remote maven access enabled for *Karaf* and external features must be locally provisioned as a *Kar* or an *RPM* before being enabled with the *Plugin Manager*.

This guide describes how to deploy and manage plugins using the *Plugin Manager*. A separate plugin developer's guide is provided for those wishing to write their own plugins.

19.1. Plugin Manager UI panel

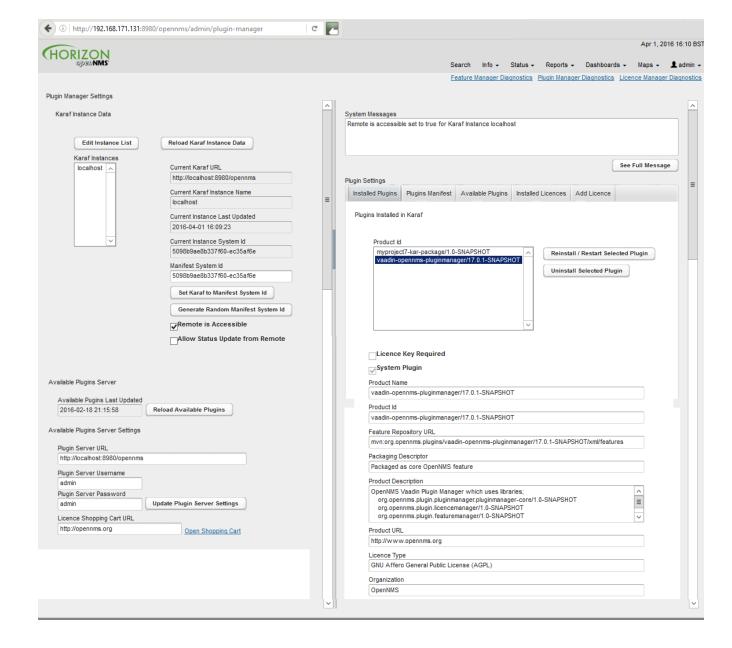
The *Plugin Manager* is accessed as an entry in the *Additional Tools* panel of the *OpenNMS Meridian Admin Gui*.

Additional Tools

Instrumentation Log Reader SNMP MIB Compiler Ops Board Configuration Surveillance Views Configuration JMX Configuration Generator OpenNMS Plugin Manager

The Plugin Manager administration page is split into six main areas as illustrated below.

- 1. Top Left is the *Karaf* Instance data panel which lists the *Karaf* instances known to the *Plugin Manager*. When a *Karaf* instance is selected, the data on the rest of the page refers to the selected instance.
- 2. Bottom Left is the *Available Plugins Server Panel* which is used to set the address and passwords to access the *Available Plugins Server* and / or the list of locally available plugins provided by a *Kar* or *RPM*.
- 3. Top Right, just below the main *OpenNMS Meridian* menu bar are links to three diagnostic pages which can help test the *ReST* interface to remote *Karaf* Instances.
- 4. Middle Right is a messages panel which reports the status of any operations. If an operation fails, the full error message can be viewed by pressing the error message button.
- 5. Bottom Right is a tabbed panel which reflects the status of the plugins and licences installed in the *Karaf* instance selected by the *Karaf* Instance data panel.



19.2. Setting Karaf Instance Data

The *Karaf* instances known to the *Plugin Manager* are listed in the *Karaf* Instance data panel. Localhost refers to the local *OpenNMS Meridian* server and is always an option in the panel. The *Karaf* instance data is persisted locally and should be refreshed from remote sources using the reload *Karaf* instance data button before changes are made.



Please note that the Localhost configuration in the *Plugin Manager* by default uses admin for both the username and the password. This will not work in a production OpenNMS where you have changed the admin user password. You should edit the Localhost configuration using the edit instance list button to match your local configuration)

Each *Karaf* instance must have a unique system id which is used to update its configuration and also to validate its licences. The system id it must be unique and included a checksum. A new random system id can be generated for a *Karaf* instance using a button on the panel.

In most situations the remote Karaf instance can be accessed from the OpenNMS Meridian Plugin

Manager. However in many cases, the remote *Karaf* will be behind a firewall in which case it must initiate the communications to request its configuration and supply an update on its status.

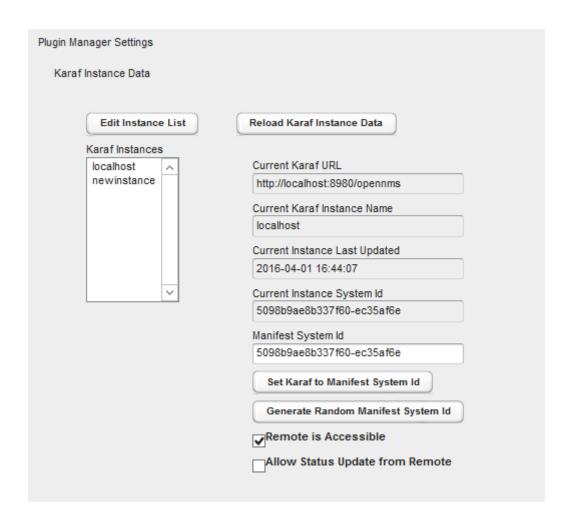
The Remote is Accessible field tells the *Plugin Manager* which mode of operation is in use.



Remote request of configuration is not yet fully implemented and will be completed in a future release.

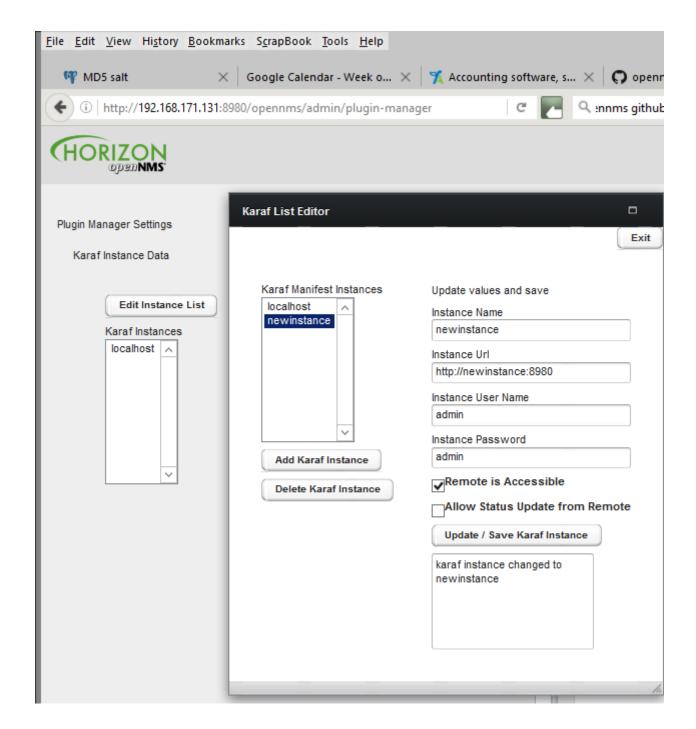
Table 105. Karaf Instance Fields

| Field Name | Description |
|---------------------------------|---|
| Instance Name | host Name of the <i>Karaf</i> instance |
| Karaf URL | URL used to access the <i>Karaf Plugin Manager</i> ReST API |
| Current Instance System ID | The system ID currently installed in the <i>Karaf</i> system |
| Manifest System ID | The system ID to be provisioned in the <i>Karaf</i> system |
| Remote is Accessible | If ticked 'true', the <i>Plugin Manager</i> will try and contact the remote <i>Karaf</i> instance using the URL. If not ticked (i.e. false), the remote <i>Karaf</i> instance must request its configuration. |
| Allow Status Update from Remote | Allow the remote <i>Karaf</i> instance to request an update to its remote configuration from the locally held manifest and at the same time to update its status. |



19.3. Manually adding a managed Karaf instance

The list of *Karaf* instances can be modified using the *Karaf* instance editor illustrated below. The same fields apply as above.



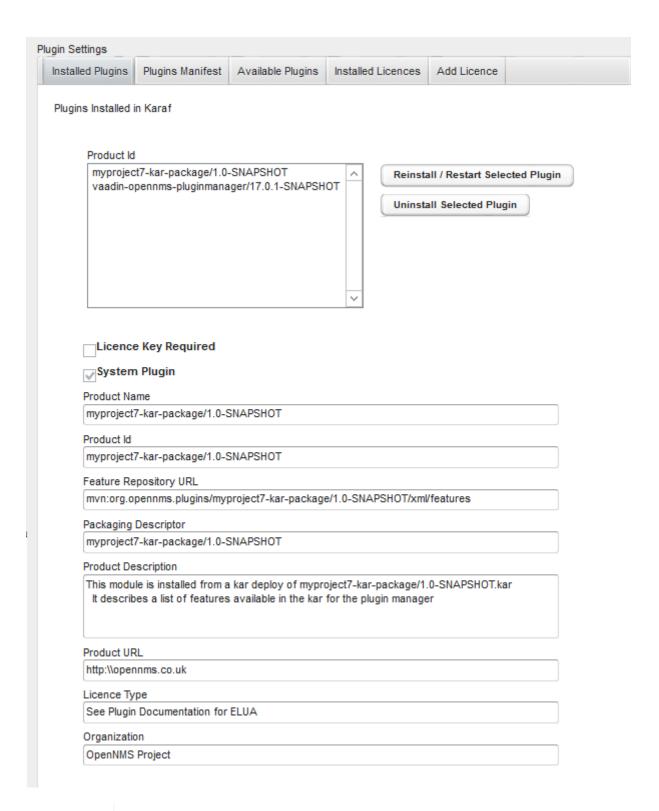
19.4. Installed Plugins

Under plugin settings, the Installed Plugins tab lists which plugins are currently installed in the *Karaf* instance selected in the *Karaf* instance data panel. System Plugins cannot be uninstalled through the UI. (The *Plugin Manager* is itself a system plugin). Non-system plugins can be reinstalled or removed from the system. Each plugin has metadata associated with it which is used to identify and describe the plugin.

Table 106. Plugin Metadata Fields

| Plugin Metadata | Description |
|-----------------|--|
| Product ID | The unique key used to identify the name and version of the feature. (Same as <i>Karaf</i> Feature Name/Version) |

| Plugin Metadata | Description |
|------------------------|---|
| Licence Key Required | If true (ticked), this plugin needs a licence key to start |
| Licence Validated | If a licence key is required, a green text label will indicate if the licence has been installed and validated. Otherwise a red text label will indicate an invalid licence |
| System Plugin | If true (ticked) this is a system plugin and cannot be removed. |
| Packaging Descriptor | This describes the packaging mechanism by which the feature was delivered. This will refer to a Kar if the feature was manually installed as a Kar/RPM on the host server. |
| Feature Repository URL | The URL identifying the feature repository (Same as <i>Karaf</i> Feature Repository URL) |
| Product Description | A textual description of the functionality provided by the plugin. |
| Product URL | A URL to point to the plugin's documentation / web site |
| licence Type | A description of the licence applied to the plugin (May be GPL if the plugin is not subject to an ELUA) |
| Organisation | The organisation issuing the plugin and/or licence. |





The installed plugins tab shows the data retrieved the last time the Reload Karaf Instance data button was pressed. (This allow us to maintain a record of offline *Karaf* instances). However it also means that the localhost data may not be up to date with the local *Karaf* instance. You should always reload to get the accurate picture of what is currently installed.

19.5. Available Plugins Server

The Plugin Manager obtains a list of available plugins from the Available Plugin's server.

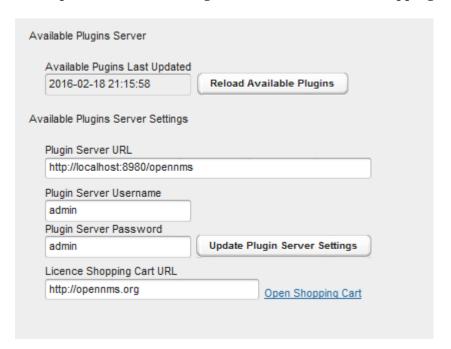
Available Plugin's server can be part of an externally hosted plugin shopping cart or it can simply be

a url serving the internal list of available plugins as described in the section on Internal Plugins.

In order for externally downloaded plugins to be installed, the *Available Plugin's server* must have a related maven repository from which *Karaf* can download the feature. By default feature download is not enabled in *OpenNMS Meridian*. To enable *Karaf* external feature download, the address of the maven repository should be entered in the org.ops4j.pax.url.mvn.cfg file in the *OpenNMS Meridian* /etc directory.

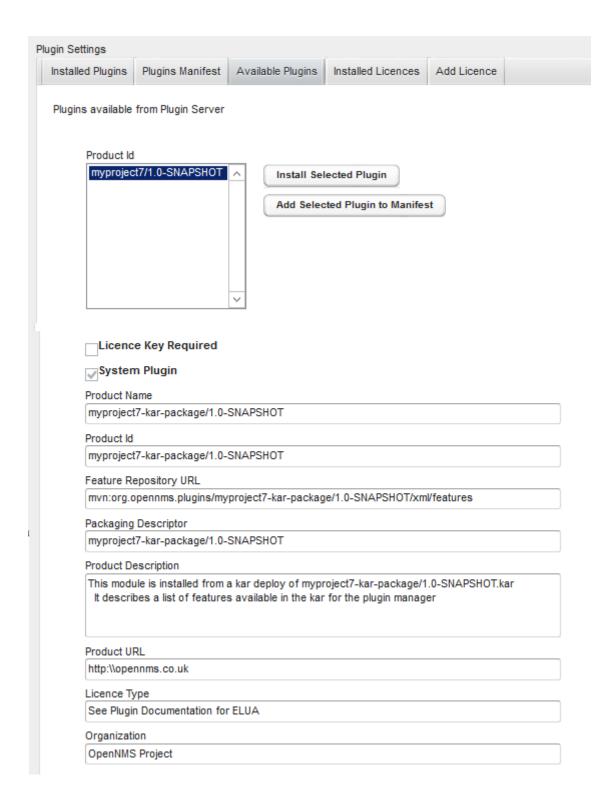
Alternatively the *Plugin Manager* can list the available plugins which have been installed on the local machine as bundled Plugin Kar's (using the *Karaf* Kar deploy mechanism) along with any internal plugins bundled with *OpenNMS Meridian*. In this case, the *Plugin Server URL* should be pointed at http://localhost:8980/opennms.

The admin username and passwords are used to access the *Available Plugins Server*. If a shopping cart is provided for obtaining licences, the URL of the shopping cart should be filled in.



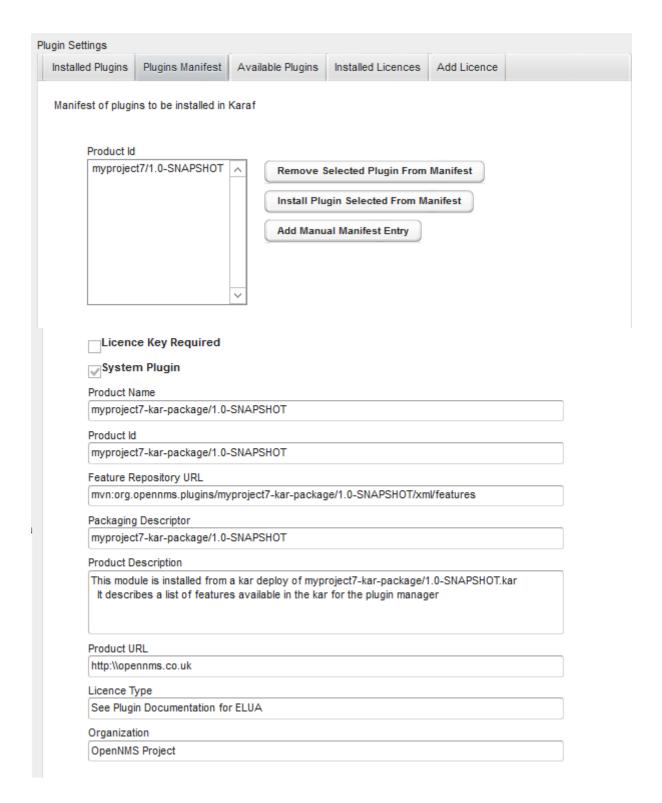
19.6. Installing Available Plugins

The Available Plugins panel list the plugins which are available and listed by the Available Plugins server. These can be directly installed into the selected *Karaf* instance or can be posted to a manifest for later installation. If a plugin is installed, the system will try and start it. However if a corresponding licence is required and not installed, the features will be loaded but not started. You must restart the feature if you later install a licence key.



19.7. Plugins Manifest

The Plugins Manifest for a given *Karaf* instance lists the target plugins which the *Karaf* instance should install when it next contacts the licence manager. If the *Plugin Manager* can communicate with the remote server, then a manifest can be selected for installation. A manual manifest entry can also be created for a feature. This can be used to install features which are not listed in the Available Features list.



19.8. Installed Licences Panel

Each licence has a licence ID which is the *Karaf* feature ID of the feature to which the licence refers. Many licences can be installed on a system but only one licence string is allowed per feature ID.

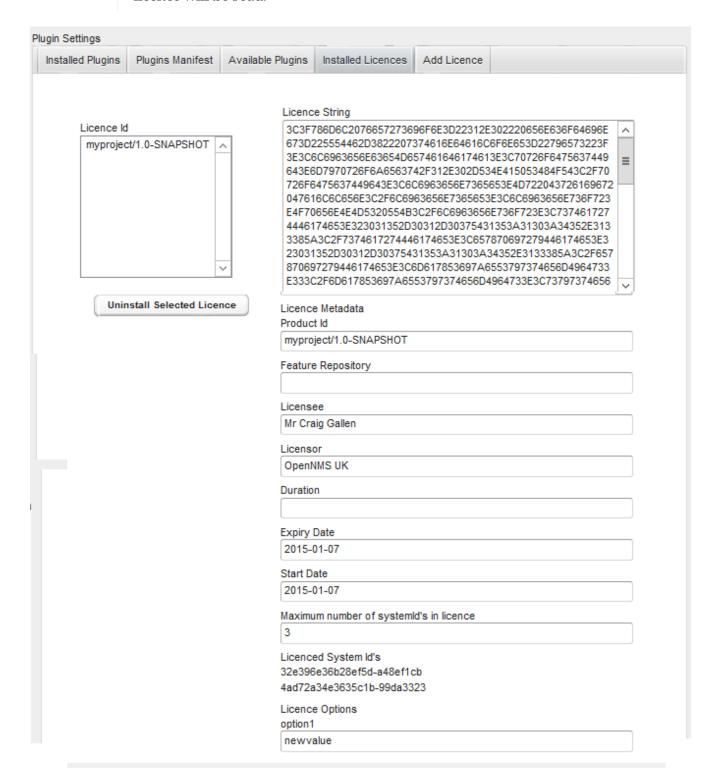
Licence Strings are used to validate that a particular feature can be run on a given *Karaf* instance. The *Plugin Manager* will not allow a feature to run if it's licence cannot be validated using a private key encoded in the feature bundle.

Licences are associated with specific Product ID's and specific *Karaf* instances. Several *Karaf* instances can be listed in a licence allowing a feature to run on more than one system using the

same licence. When a licence is installed, the licence metadata is decoded and displayed.



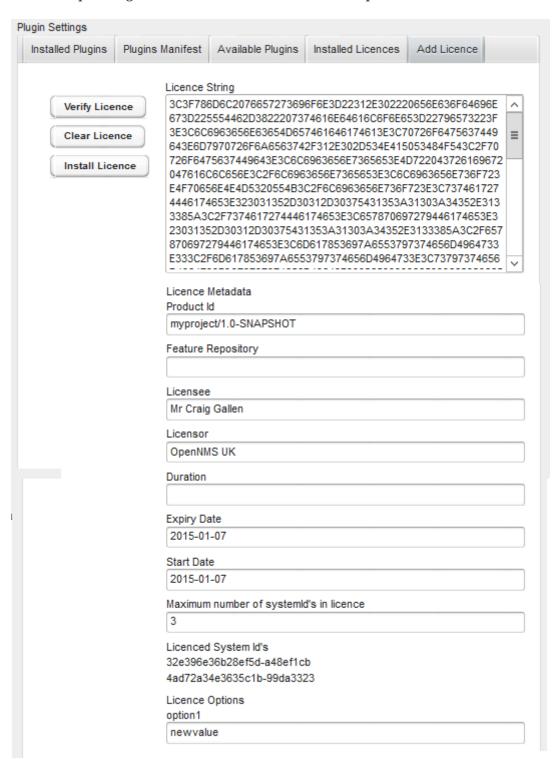
A licence may be installed before or after its associated feature is installed. If a licence is installed after the feature the feature must be restarted before the licence will be read.



19.9. Adding a New Licence

New licences are added using the add licence panel. Licences are obtained from the *App Store* where they can be generated by a user for a given set of system id's.

A licence must be copied (cut and paste) from the app store into the add licence panel. The Validate licence button should be used to check the licence has been installed correctly. Please note that this just checks the integrity of the licence string. A licence is only authenticated once it is installed and the corresponding feature bundle checks it on start-up.



19.10. Installing Internal Plugins

OpenNMS Meridian is packaged with an internal repository of plugins which are shipped with the *OpenNMS Meridian* distribution. These plugins can be installed in the local *OpenNMS Meridian Karaf* instance and activated by a user using the *Plugin Manager* in the same way it could be used to download and install external plugins.

The internal-plugin-descriptor feature maintains a list of internal plugins which are packaged with OpenNMS Meridian. This list of internal plugins can be accessed by the Plugin Manager by setting the Available Plugins Server to point to the local OpenNMS Meridian instance. To do this set Plugin Server URL to the address of the local OpenNMS Meridian (i.e. http:\\localhost:8980\opennms) and set the Plugin Server Username and Plugin Server Password to match the OpenNMS Meridian ReST or admin username and password.

Clicking Reload available plugins will then add the list of available internal plugins to the Available Plugins Tab where they can be installed and started by the user as described previously.

The internal plugins included with this *OpenNMS Meridian* release are documented in the next section.

Chapter 20. Internal Plugins

20.1. Internal Plugins supplied with OpenNMS Meridian

OpenNMS Meridian includes a number of plugins which can be installed by the Plugin Manager UI or directly from the *Karaf* consol. Plugins are simply *Karaf* features which have additional metadata describing the Plugin and possibly defining that the Plugin also needs a licence installed to run.

Once installed, the plugins will always start when OpenNMS is restarted. If the plugins appear not to be working properly, you should check the /data/log/karaf.log file for problems.

Each internal plugin supplied with *OpenNMS Meridian* is described in its own section below.

20.2. Installing Plugins with the Karaf Consol

The easiest way to install a plugin is to use the Plugin Manager UI described in the Plugin Manager section. However plugins can also be installed using the *Karaf* consol. To use the *Karaf* consol, you need to open the karaf command prompt using

```
ssh -p 8101 admin@localhost
(or ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no if no host
checking is wanted)
```

To install or remove a feature in *Karaf* use

```
karaf@root> features:install <feature name>
karaf@root> features:uninstall <feature name>
```

You can see which plugins are installed using

```
karaf@root> product-reg:list
```

20.3. Alarm Change Notifier Plugin

The *Alarm Change Notifier Plugin* generates new OpenNMS events corresponding to changes in alarms The new events are defined in the <opennms home>/etc/events/AlarmChangeNotifierEvents.xml file

These events contain a json copy of the database table before changes in %parm[oldalarmvalues]% and after changes in %parm[newalarmvalues]%. (New Alarm events do not contain %parm[oldalarmvalues] and Alarm Deleted events do not contain %parm[newalarmvalues]%)

%parm[alarmid]% contains the alarmid of the alarm which has changed

The generated event itself references copies of the nodeid, interface and service contained in the original alarm. This way the alarm change events are associated with the original source of the alarm.

Alarm change events have a severity of normal since they only reflect changes to the alarm.

Events from the alarm-change-notifier are also used by the opennms-es-rest plugin to send alarm history to Elasticsearch

The table below lists the parameters included with each type of Alarm Change Event. Parameters are listed in the %parm[xxx]% format which is used to reference them in AlarmChangeNotifierEvents.xml

To simplify searching and visualisation, specific parameter values are also added for each alarm change event type. These additional values are described in the table below.

| Alarm Change Event Type | UEI | Additional Parameters |
|-------------------------------|--|---|
| New Alarm Created | uei.opennms.org/plugin/AlarmChang eNotificationEvent/NewAlarmCreated | %parm[alarmid]% %parm[newalarmvalues]% |
| Alarm Severity Changed | uei.opennms.org/plugin/AlarmChang eNotificationEvent/AlarmSeverityCha nged | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% %parm[severity]% %parm[oldseverity]% |
| Alarm Cleared | uei.opennms.org/plugin/AlarmChang eNotificationEvent/AlarmCleared | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% |
| Alarm Deleted | uei.opennms.org/plugin/AlarmChang eNotificationEvent/AlarmDeleted | %parm[alarmid]% %parm[oldalarmvalues]% |
| Alarm Changed | uei.opennms.org/plugin/AlarmChang eNotificationEvent/AlarmChanged | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% |
| Alarm Acknowledg ed | uei.opennms.org/plugin/AlarmChang eNotificationEvent/AlarmAcknowledg ed | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% %parm[alarmid]% %parm[alarmacktime]% %parm[alarmackuser]% |
| Alarm UnAcknowle dged | uei.opennms.org/plugin/AlarmChang eNotificationEvent/AlarmUnAcknowl edged | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% |
| Alarm Suppressed | uei.opennms.org/plugin/AlarmChang eNotificationEvent/AlarmSuppressed | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% %parm[suppressedtime]% %parm[suppresseduntil]% %parm[suppresseduser]% |
| Alarm UnSuppress ed | uei.opennms.org/plugin/AlarmChang eNotificationEvent/AlarmUnSuppress ed | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% |

| Alarm Change Event Type | UEI | Additional Parameters |
|----------------------------------|--|--|
| TroubleTick etStateChan ge | uei.opennms.org/plugin/AlarmChang eNotificationEvent/TroubleTicketStat eChange | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% %parm[tticketid]% %parm[tticketstate]% |
| Sticky Memo Added | uei.opennms.org/plugin/AlarmChang eNotificationEvent/StickyMemoAdded | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% %parm[stickymemo]% |
| Sticky Memo Update | uei.opennms.org/plugin/AlarmChang eNotificationEvent/StickyMemoUpdat e | %parm[alarmid]% %parm[oldalarmvalues]% %parm[newalarmvalues]% %parm[oldalarmvalues]% %parm[stickymemo]% %parm[author]% %parm[body]% %parm[memovalues]% |
| Journal Memo Update | uei.opennms.org/plugin/AlarmChang eNotificationEvent/JournalMemoUpd ate | %parm[alarmid]% %parm[newalarmvalues]% %parm[oldalarmvalues]% %parm[author]% %parm[body]% %parm[reductionkey]% %parm[memovalues]% |

20.4. Elasticsearch ReST plugin

The *Elasticsearch ReST plugin* provides an interface to forward events, alarms and alarm change events generated by the *Alarm Change Notifier Plugin* to *Elasticsearch* (https://github.com/elastic/elasticsearch). The events and alarms in *Elasticsearch* can be used for indexing, long time archival, plotting with *Grafana* and browsing with *Kibana*.

This plugin uses the *Elasticsearch* ReST interface and can interact with cloud-hosted Elasticsearch instances. The interface has been tested with *Elasticsearch* 2.4, 5.0, and 5.1.



If you use *Kibana*, make sure you are using the version that is compatible with your version of *Elasticsearch*.

The *Elasticsearch ReST plugin* uses the Jest library (https://github.com/searchbox-io/Jest) to access the *Elasticsearch* ReST interface.

20.4.1. Configuration

Configuration is held in:

/etc/org.opennms.plugin.elasticsearch.rest.forwarder.cfg

With the following properties (defaults shown will be used if file is not present)

| Parameter | Default Value | Requi red | Description |
|------------------------------|---------------------------|--------------|--|
| elasticsearchUr l | http://localh ost:9200 | optio nal | URL of <i>Elasticsearch</i> ReST interface. This value can also contain a comma-separated list of URLs that will be used in round-robin fashion for increased scalability. |
| esusername | | optio nal | Username to access <i>Elasticsearch</i> . |
| espassword | | optio nal | Password to access <i>Elasticsearch</i> . |
| logEventDescrip tion | true | optio nal | Whether to forward the event description field to <i>Elasticsearch</i> . It can be disabled because it contains a long text field that can be redundant with the rest of the metadata included in the event. |
| archiveRawEvent s | true | optio nal | Archive events. |
| archiveAlarms | true | optio nal | Archive alarms. |
| archiveAlarmCha ngeEvents | true | optio nal | Archive alarm change events. |
| archiveOldAlarm Values | true | optio nal | For alarm change events, we can choose to archive the detailed alarm values but this is expensive. Set false in production. |
| archiveNewAlarm Values | true | optio nal | |
| archiveAssetDat a | true | optio nal | If true The following attributes representing useful node asset fields from the node asset table are included in archived events and alarms. These are included only where the values are not null or empty strings in the table. (asset-latitude,asset-longitude,asset-region,asset-building,asset-floor,asset-room,asset-rack,asset-slot,asset-port,asset-category,asset-displaycategory,asset-notifycategory,asset-pollercategory,asset-thresholdcategory,asset-managedobjecttype,asset-managedobjectinstance,asset-manufacturer,asset-vendor,asset-modelnumber,parent-nodelabel,parent-nodeid,parent-foreignsource,parent-foreignid) |
| logAllEvents | false | optio nal | If changed to true, then archive all events even if they have not been persisted in the <i>OpenNMS Meridian</i> database. |
| retries | 0 | optio nal | The number of times to retry an <i>Elasticsearch</i> operation that fails completely. You can increase retries to avoid losing forwarded events and alarms when <i>Elasticsearch</i> is down or unreachable. |
| timeout | 5000 | optio nal | The interval between subsequent retries when a retries value greater than 1 is being used. |

| Parameter | Default Value | Requi red | Description |
|---------------|------------------|--------------|---|
| batchSize | 1 | optio nal | Increase this value to enable batch inserts into Elasticsearch. This is the maximum size of a batch of events that is sent to Elasticsearch in a single connection. |
| batchInterval | 0 | optio nal | The maximum time interval in milliseconds between batch events (recommended: 500ms) when a batchSize value greater than 1 is being used. |

Once you are sure everything is correctly configured, you can activate the *Elasticsearch* forwarder by logging into the *OSGi* console and installing the feature.

OSGi login and installation of the Elasticsearch forwarder

```
ssh admin@localhost -p 8101
features:install opennms-es-rest
```

20.4.2. Loading Historical Events

It is possible to load historical *OpenNMS Meridian* events into *Elasticsearch* from the *OpenNMS Meridian* database using a *karaf* console command. The command uses the *OpenNMS Meridian* Events ReST interface to retrieve a set number of historical events and forward them to *Elasticsearch*. Because we are using the ReST interface it is also possible to contact a remote *OpenNMS Meridian* and download its events into *Elasticsearch* by using the correct remote URL and credentials.

```
open karaf command prompt using ssh -p 8101 admin@localhost
```

To send historic events to *Elasticsearch* use a command of the form:

```
karaf> elastic-search:send-historic-events limit offset [ onms-username onms-password
onms-url use-node-label ]
```

The *mandatory* parameters are

- limit Limit of number of events to send
- offset Offset for starting list of events

(note that the limit parameter works in multiples of 10 and may send more than the limit to round to 10 events)

The following parameters are optional and will use defaults if not set

• onms-username - ReST password for opennms (default: admin)

- onms-password ReST username for opennms (default: admin)
- onms-url URL of *OpenNMS Meridian* ReST interface to retrieve events to send (default: http://localhost:8980)
- use-node-label If false local node cache will get nodelabel for nodeid. If true will use remote nodelabel (default: false)

If you are uploading events from the local machine on which you are running this command, you should use the local node cache as this supplies a number of node values including the nodelabel. If you are uploading from a remote machine you should use the remote node label and not the local node cache. Only the remote nodelabel is provided in this case.

Command examples:

```
# This retrieves 110 alarms from the local machine using the
# local node cache for node label
elastic-search:send-historic-events 100 0 admin admin http://localhost:8980 false
# This retrieves 110 alarms from the remote machine using the remote node labels
elastic-search:send-historic-events 100 0 demo demo http://demo.opennms.org true
```

20.4.3. Index Definitions

Three indices are created; one for alarms, one for alarm change events and one for raw events. Alarms and alarm change events are only saved if the alarm-change-notifier plugin is also installed to generate alarm change events from the *OpenNMS Meridian* alarms table. The index names are of the form:

```
<name>-<date>/type/id
```

For example

a) Alarms

```
opennms-alarms-2017.01/alarmdata/1823
```

b) Alarm Change Events

```
opennms-events-alarmchange-2017.01/eventdata/11549
```

c) Raw OpenNMS Meridian events (not including alarm change events)

```
opennms-events-raw-2017.01/eventdata/11549
```

20.4.4. Viewing events using Kibana Sense

Kibana Sense is a *Kibana* app which allows you to run queries directly against *Elasticsearch*. (https://www.elastic.co/guide/en/sense/current/installing.html)

If you install *Kibana Sense* you can use the following commands to view the alarms and events sent to *Elasticsearch* You should review the *Elasticsearch* ReST API documentation to understand how searches are specified. (See

https://www.elastic.co/guide/en/elasticsearch/reference/current/search.html)

Example searches to use in *Kibana Sense* (you can copy the whole contents of this panel into *Kibana Sense* as a set of examples)

```
# Search all the alarms indexes
GET /opennms-alarms-*/ search
# Get all of the alarms indexes
GET /opennms-alarms-*/
# Get a specific alarm id from the 2017.01 index
GET opennms-alarms-2017.01/alarmdata/1823
# Delete all alarm indexes
DELETE /opennms-alarms-*/
# Search all the events indexes
GET /opennms-events-*/_search
# Search all the raw events indexes
GET /opennms-events-raw*/ search
# Delete all the events indexes
DELETE /opennms-events-*/
# Get all the raw events indexes
GET /opennms-events-raw*/
# Get all the alarmchange event indexes
GET /opennms-events-alarmchange-*/
# Search all the alarm change event indexes
GET opennms-events-alarmchange-*/_search
# Get a specific alarm change event
GET opennms-events-alarmchange-2016.08/eventdata/11549
```

20.4.5. Mapping of Alarms and Events to Elasticsearch

Overview of index mapping

In OpenNMS Meridian, Alarm and Event table entries contain references to associated node, asset,

service and journal message tables. In *Elasticsearch*, we must flatten these entries into a single index entry for each insertion. Thus each index entry contains more context information than would be found in the actual *OpenNMS Meridian* event or alarm. This context information includes the associated node and asset table information which was current when (but may have changed since) the event was archived.

In the *Table of Index Mappings* below we have example alarm and event JSON entries retrieved using a sense command. The table helps illustrate how *OpenNMS Meridian* saves data in *Elasticsearch*.

Internal *Elasticsearch* fields always begin with an underscore character. The internal fields id, _index and _type are combined to give the unique identifier for an entry as described above under Index Definitions. All of the fields under _source represent the stored alarm or event (_Elasticsearch documentation refers to source entries as indexed documents). The ID of each event is included in the _source id field and also duplicated in the internal _id.

Events in the *OpenNMS Meridian* events table (i.e. those corresponding to logs or traps) are copied directly to the opennms-events-raw- indexes. In *OpenNMS Meridian* alarms and events can contain parameters which are key-value pairs referencing additional data stored when the event or alarm is created. In *Elasticsearch* these parameters are always stored in separate fields in the index with names beginning with p_

Alarm change events created by the *Alarm Change Notifier Plugin* have an identical format to raw events but are only copied to the opennms-events-alarmchange- indexes. These alarm change events are also used to change the state of alarms in the opennms-alarms- indexes. Thus alarm entries in the opennms-alarms- indexes reflect the current state of alarms as notified by *OpenNMS Meridian* through alarm change events.

The parameters included with each type of Alarm Change Event are listed in the *Alarm Change Notifier Plugin* section. Each parameter in the index will have a p_ prefix (ie. %parm[newalarmvalues]% becomes p_newalarmvalues).

Alarms and Events have severity fields defined as integers (long) and also corresponding severity text fields which give the text equivalent (Critical, Major, Minor, Normal, Cleared).

Additional Alarm Fields

The id of each alarm is included in the _source alarmid field and also duplicated in the internal _id reference for the alarms index. Alarm Change Events reference their associated alarm using the p_alarmid parameter. To make it easier to search for alarm change events associated with the same alarm, alarms also have a _source p_alarmid parameter which matches alarmid. Thus we should be able to search for an alarm in the opennms-alarms index and find its complete lifecycle from alarm raise to deletion in the opennms-events-alarmchange index.

The alarms index is enriched with additional data to allow the alarm entries to be used in SLA calculations.

| Additional Alarm Fields | description |
|----------------------------|--|
| alarmackduration | Calculated time in milliseconds from first event which created the alarm to the latest alarm acknowledgement. |
| alarmclearduratio n | Calculated time in milliseconds from first event which created the alarm to the latest alarm clear. |
| initialseverity | The final state of any given alarm in an alarm index should be cleared and deleted. Therefore we also include an initial severity. |
| initialseverity_text | The initial severity as a text field. |

Table of Index Mapping

The following table describes the mapping of simple *OpenNMS Meridian* events to the Raw Events Index and the mapping of Alarm Change Events to the Alarm Change Events index and to the Alarms index. Note that fields that begin with an underscore (_) are internal to Elasticsearch.

| Alarm Index Fields | | Event Index Fields (Alarm change and raw events) | | Description | |
|---|-------------------|--|---|----------------|---|
| Example Alarm JSON | Alarm Field | Event Field | Example Event JSON | Ty pe | Description |
| { | { | { | { | | |
| "_index": "opennms-alarms-2017.03", | "_index ": | "_inde x": | "_index": "opennms- events-alarmchange- 2017.03", | str in g | _index is the index in which this alarm or event is stored. |
| "_type": "alarmdata", | "_type": | "_type ": | "_type": "eventdata", | str in g | _type either alarmdata or eventdata |
| "_id": "31", | "_id": | "_id": | "_id": "1110", | str in g | _id field matches the event or alarm ID, if present. |
| "_score": 1, | "_score ": | "_scor e": | "_score": 1, | lo ng | Internal Elasticsearch ranking of the search result. |
| "_source": { | "_sourc e": | "_sour ce": | "_source": { | str in g | _source contains the data of the index entry. |
| "@timestamp": "2017-03- 03T12:44:21.210Z", | "@time stamp": | | "@timestamp": "2017-03- 02T15:20:56.861Z", | da te | For Alarms, <code>@timestamp</code> is alarm creation time based on the first event time. For Events, <code>@timestamp</code> is event time from <code>event.getTime()</code> . |

| Alarm Index Fields | | | Index Fields (Alarm e and raw events) | Description | |
|--------------------|---------|-----------------------------------|---|----------------|--|
| "dom": "3", | "dom": | "dom" | "dom": "2", | lo ng | Day of month from @timestamp. |
| "dow": "6", | "dow": | "dow": | "dow": "5", | lo ng | Day of week from <pre>@timestamp.</pre> |
| "hour": "12", | "hour": | "hour" | "hour": "15", | lo ng | Hour of day from @timestamp. |
| | | "event descr" : | "eventdescr": "Alarm 30 Cleared",</a | in | Event description. |
| | | "event severi ty": | "eventseverity": "3", | lo ng | Event severity. <i>Alarm Change Events:</i> All events have severity normal. |
| | | "event severi ty_text ": | "eventseverity_text": "Normal", | str in g | Text representation of severity value. |
| | | | "eventsource": "AlarmChangeNotifier", | str in g | OpenNMS event source. Alarm Change Events: All events have the event source AlarmChangeNotifier. |
| | | "event uei": | "eventuei": "uei.opennms.org/plugin/ AlarmChangeNotificatio nEvent/AlarmCleared", | | OpenNMS universal event identifier (UEI) of the event. |
| | | "id": | "id": "1110", | str in g | Event ID. |
| | | "interf ace": | "interface": "127.0.0.1", | str in g | IP address of the event. |
| | | "ipadd r": | "ipaddr": "/127.0.0.1", | str in g | IP address of the event. |
| | | "logm sg": | "logmsg": "Alarm 30 Cleared",</a | | Log message of the event. Alarm Change Events: Log messages contain a link to the alarm. |

| Alarm Index Fields | Index Fields (Alarm e and raw events) | Des | Description | | |
|--------------------|--|-----|-------------------------------|--|--|
| | "logmsgdest": "logndisplay", | | Log Destination of the Event. | | |

| Alarm Index Fields | | Index Fields (Alarm e and raw events) | Des | scription |
|--------------------|------------------------|---|----------|---|
| | "p_ne walar mvalu es": | \"systemid\":\"00000000000000000000000000000000\",+ \"suppresseduntil\":\"201 7-03-02T14:24:59.282Z\",+ \"description\":\"SNM P data collection on interface 127.0.0.1\\n failed.<\\/p>\", \"mouseovertext\":null, \"x733probablecause\":0, \"lasteventid\":1072, \"lasteventtime\":\"2017- 03-02T14:24:59.282Z\", \"managedobjectinstance \":null, \"qosalarmstate\":null, \"ipaddr\":\"127.0.0.1\", \"alarmacktime\":null, \"ipaddr\":\"127.0.0.1\", \"alarmackuser\":null, \"nodeid\":88, \"firsteventtime\":\"2017- 03-02T14:24:59.282Z\", \"severity\":2, \"ifindex\":null, \"nodeid\":88, \"firsteventtime\":\"2017- 03-02T14:24:59.282Z\", \"severity\":2, \"ifindex\":null, \"nodeid\":8NMP data collection on interface 127.0.0.1 failed with Unexpected exception when collecting SNMP data for interface 127.0.0.1 at location Default.'.\", \"tticketid\":null, \"firstautomationtime\":n ull, \"clearkey\":null, \"managedobjecttype\":n ull, \"counter\":1, \"applicationdn\":null, \"operinstruct\":null, \"operinstruct\":null, \"operinstruct\":null, \"operinstruct\":null, \"operinstruct\":null, \"operinstruct\":null, \"operinstruct\":null, \"operinstruct\":null, \"ossprimarykey\":null, \"ossprimarykey\":null, \"ossprimarykey\":null, \"ossprimarykey\":null, \"ossprimarykey\":null, | str in g | Alarm and event parameters are keyvalue pairs which can be associated with alarms or events. All parameters in Alarms or Events are stored in Elasticsearch in separate index fields with names beginning with p Alarm Change Events: Parameters p_oldalarmvalues and p_newalarmvalue contain a JSON string representing the alarm fields before and after the Alarm change respectively. The p_newalarmvalue values are copied into the alarm index of the corresponding alarm (given by alarmid in p_newalarmvalue and by p_alarmid). |
| | | \"stickymemo\":null, \"tticketstate\":null, | | 335 |

| Alarm Index Fields | | Event Index Fields (Alarm change and raw events) | | | Description | | |
|---|-------------------------------|--|--|----------------|---|--|--|
| | | _ | "p_oldalarmvalues": "{ }", | str in g | See p_newalarmvalues. | | |
| | | "p_old severi ty": | "p_oldseverity": "5", | lo ng | Alarm Change Events: Contains the old severity of the alarm before this alarm change event. | | |
| "alarmackduration": "2132249", | "alarm ackdur ation": | | | lo ng | Time in milliseconds from first event which created the alarm to the latest alarm acknowledgement. | | |
| "alarmacktime": "2017-03-03T13:19:53.351Z", | "alarm acktim e": | _ | "p_alarmacktime": "2017-03-03T13:19:53.351Z", | da te | AlarmChangeNotification Event/AlarmAcknowledge d Events: Time that the alarm was acknowledged. | | |
| "alarmackuser": "admin", | "alarm ackuser ": | _ | "p_alarmackuser": "admin", | | AlarmChangeNotification Event/AlarmAcknowledge d Events: Name of the user who acknowledged the alarm. | | |
| "alarmclearduration": "2175014" | "alarm cleardu ration": | | | lo ng | Time in milliseconds from first event which created the alarm to the latest alarm clear. | | |
| "alarmcleartime": "2017-03-03T13:20:36.224Z", | "alarm clearti me": | "p_ala rmcle artime ": | "p_alarmcleartime": "2017-03- 03T13:20:36.224Z", | da te | AlarmChangeNotification Event/AlarmClear Events: Time that the alarm was cleared. | | |
| "alarmid": "31", | "alarmi d": | "p_ala rmid": | "p_alarmid": "30", | str in g | Alarm Change Events: The alarm ID of the alarm that has changed. | | |
| "alarmtype": "1", | "alarmt ype": | "p_ala rmtyp e": | "p_alarmtype": "1", | str in g | Alarm Change Events: Corresponds to the alarm's type. | | |
| "applicationdn": null, | "applic ationdn ": | | | str in g | | | |

| Alarm Index Fields | | | Index Fields (Alarm e and raw events) | Description | |
|---|---------------------------|---------------------------|---|----------------|--|
| "asset-category": "Power", | "asset- categor y": | "asset- catego ry": | "asset-category": "Power", | str in g | All asset_ entries correspond to fields in the Asset Table of the node referenced in the event. These fields are only present if populated in the asset table. |
| "asset-building": "55", | "asset- buildin g": | "asset- buildi ng": | "asset-building": "55", | str in g | |
| "asset-room": "F201", | "asset- room": | "asset- room" : | "asset-room": "F201", | str in g | |
| "asset-floor": "Gnd", | "asset- floor": | "asset- floor": | "asset-floor": "Gnd", | str in g | |
| "asset-rack": "2101", | "asset- rack": | "asset- rack": | "asset-rack": "2101", | str in g | |
| "categories": "", | "catego ries": | "categ ories": | "categories": "", | str in g | categories corresponds to node categories table. This is a comma- separated list of categories associated with this node ID. This field is indexed so separate values can be searched. |
| "clearkey": null, | "cleark ey": | | | str in g | |
| "counter": "1", | "counte r": | | | str in g | |
| "description": "SNMP data collection on interface 127.0.0.1\n failed.", | "descri ption": | | | str in g | |
| "eventuei": "uei.opennms.org/nodes/dataCollectionFailed", | | _ | "p_eventuei": "uei.opennms.org/nodes/dataCollectionFailed", | str in g | Alarm Change Events: Corresponds to the alarm's event UEI. |

| Alarm Index Fields | | Event Index Fields (Alarm change and raw events) | | | Description | | |
|---|--------------------------------------|--|---|----------------|--|--|--|
| "firstautomationtime": null, | "firstau tomatio ntime": | | | da te | | | |
| "firsteventtime": "2017-03-03T12:44:21.210Z", | "firstev enttime ": | | | da te | | | |
| "foreignid": "1488375237814", | "foreig nid": | "forei gnid": | "foreignid": "1488375237814", | str in g | Foreign ID of the node associated with the alarm or event. | | |
| "foreignsource": "LocalTest", | "foreig nsourc e": | "forei gnsou rce": | "foreignsource": "LocalTest", | str in g | Foreign source of the node associated with alarm or event. | | |
| "ifindex": null, | "ifinde x": | | | str in g | | | |
| "ipaddr": "127.0.0.1", | "ipaddr ": | | | str in g | | | |
| "lastautomationtime": null, | "lastaut omatio ntime": | | | | | | |
| "lasteventid": "1112", | "lasteve ntid": | | | str in g | | | |
| "lasteventtime": "2017- 03-03T12:44:21.210Z", | "lasteve nttime" | | | | | | |
| "logmsg": "SNMP data collection on interface 127.0.0.1 failed with 'Unexpected exception when collecting SNMP data for interface 127.0.0.1 at location Default.'.", | "logms g": | "p_log msg": | "p_logmsg": "SNMP data collection on interface 127.0.0.1 failed with 'Unexpected exception when collecting SNMP data for interface 127.0.0.1 at location Default.'.", | str in g | | | |
| "managedobjectinstance": null, | "manag edobjec tinstan ce": | | | str in g | | | |
| "managedobjecttype": null, | "manag edobjec ttype": | | | str in g | | | |

| Alarm Index Fields | | Event Index Fields (Alarm change and raw events) | | Description | |
|---|----------------------------|--|---|----------------|--|
| "mouseovertext": null, | "mouse overtex t": | | | str in g | |
| "nodeid": "88", | "nodeid ": | "nodei d": | "nodeid": "88", | str in g | Node ID of the node associated with the alarm or event. |
| "nodelabel": "localhost", | "nodela bel": | "nodel abel": | "nodelabel": "localhost", | str in g | Node label of the node associated with the alarm or event. |
| "nodesyslocation": "Unknown (edit /etc/snmp/snmpd.conf)", | "nodes yslocati on": | "node sysloc ation": | "nodesyslocation": "Unknown (edit /etc/snmp/snmpd.conf)", | str in g | SNMP syslocation of the node associated with the alarm or event. |
| "nodesysname": "localhost.localdomain", | "nodes ysname ": | "node sysna me": | "nodesysname": "localhost.localdomain", | str in g | SNMP sysname of the node associated with the alarm or event. |
| "operatingsystem": null, | "operat ingsyst em": | | | str in g | |
| "operinstruct": null, | "operin struct": | | | str in g | |
| "ossprimarykey": null, | "osspri maryke y": | | | str in g | |
| "p_alarmid": "31", | "p_alar mid": | | | str in g | The Elasticsearch alarms index has a field p_alarmid which corresponds to the alarmid of the alarm and also the p_alarmid field in Alarm Change Events. This allows Alarm and Alarm Change Event indexes to be easily searched together for all Alarm Change Events corresponding to an alarm. |

| Alarm Index Fields | | Event Index Fields (Alarm change and raw events) | | Description | |
|--|--------------------------|--|--|----------------|---|
| "p_reason": "Unexpected exception when collecting SNMP data for interface 127.0.0.1 at location Default.", | "p_reas on": | | | str in g | All parameters in Alarms or Events are stored in Elasticsearch in separate index fields with names beginning with p p_reason is an example parameter injected by the uei.opennms.org/nodes/dataCollectionFailed event in OpenNMS. |
| "qosalarmstate": null, | "qosala rmstate ": | | | str in g | |
| "reductionkey": "uei.opennms.org/nodes/ dataCollectionFailed::88", | ionkey" | _ | "p_reductionkey": "uei.opennms.org/nodes/ dataCollectionFailed::88", | in | Alarm Change Events: Corresponds to alarm reductionkey. |
| "serviceid": "5", | "servic eid": | "p_ser viceid ": | "p_serviceid": "5" | str in g | Alarm Change Events: Corresponds to the alarm's service ID. |
| "severity": "2", | "severit y": | "p_ala rmsev erity": | "p_alarmseverity": "2", | str in g | Alarm Change Events: Corresponds to the alarm's severity. |
| "severity_text": "Cleared", | "severit y_text": | | | str in g | |

| Alarm Index Fields | | Event Index Fields (Alarm change and raw events) | | | Description | |
|---|----------------------------|--|---|----------------|--|--|
| "stickymemo": null, | "sticky memo": | _ | "p_stickymemo": null, | str in g | AlarmChangeNotification Event/StickyMemoAdded Events: Content of current sticky memo for the alarm. AlarmChangeNotification Event/StickyMemoUpdate Events: These events have parameters: * p_author: author of stickymemo * p_body: content of sticky memo AlarmChangeNotification Event/JournalMemoUpda te Events: These events have parameters: * p_author: user who authored the memo * p_body: content of the memo * p_reductionkey: reduction key associated with memo (corresponds to alarm reduction key) Note that journal memos do not have an entry in the alarm index but are only referenced by reduction key. | |
| "suppressedtime": "2017-03-03T12:44:21.210Z", | | presse | "p_suppressedtime": "2017-03- 02T14:24:59.282Z", | da te | AlarmChangeNotification Event/AlarmSuppressed Events: Corresponds to the alarm's suppressed time. | |
| "suppresseduntil": "2017-03-03T12:44:21.210Z", | "suppre ssedunt il": | | "p_suppresseduntil": "2017-03- 02T14:24:59.282Z", | da te | AlarmChangeNotification Event/AlarmSuppressed Events: Corresponds to the alarm's suppressed until time. | |
| "suppresseduser": null, | "suppre sseduse r": | | "p_suppresseduser": null, | str in g | AlarmChangeNotification Event/AlarmSuppressed Events: Corresponds to the alarm's suppressed user. | |
| "systemid": "00000000- 0000-0000-0000- 000000000000 | "system id": | "p_sys temid" : | "p_systemid": "000000000- 0000-0000-0000- 00000000000 | str in g | Alarm Change Events: Corresponds to the alarm's system ID. | |

| Alarm Index Fields | | Event Index Fields (Alarm change and raw events) | | | Description | | |
|---------------------------|------------------------------|--|-------------------------|----------------|---|--|--|
| "tticketid": null, | "p_ttick etid": | "p_ttic ketid": | "p_tticketid": null, | str in g | AlarmChangeNotification Event/TroubleTicketState Change Events: Corresponds to the alarm's trouble ticket ID. | | |
| "tticketstate": null, | _ | "p_ttic ketstat e": | "p_tticketstate": null, | str in g | AlarmChangeNotification Event/TroubleTicketState Change Events: Corresponds to the alarm's trouble ticket state. | | |
| "x733alarmtype": null, | "x733al armtyp e": | | | str in g | | | |
| "x733probablecause": "0", | "x733p robable cause": | | | str in g | | | |
| } | } | } | } | | | | |

Chapter 21. Special Cases and Workarounds

21.1. Overriding SNMP Client Behavior

By default, the SNMP subsystem in OpenNMS Meridian does not treat *any* RFC 3416 error-status as fatal. Instead, it will attempt to continue the request, if possible. However, only a subset of errors will cause OpenNMS Meridian's SNMP client to attempt retries. The default SNMP error-status handling behavior is as follows:

Table 107. Default SNMP Error Status Behavior

| error-status | Fatal ? | Retry ? |
|-------------------------|------------|---------|
| noError(0) | false | false |
| tooBig(1) | false | true |
| noSuchName(2) | false | true |
| badValue(3) | false | false |
| readOnly(4) | false | false |
| genErr(5) | false | true |
| noAccess(6) | false | true |
| wrongType(7) | false | false |
| wrongLength(8) | false | false |
| wrongEncoding(9) | false | false |
| wrongValue(10) | false | false |
| noCreation(11) | false | false |
| inconsistentValue(12) | false | false |
| resourceUnavailable(13) | false | false |
| commitFailed(14) | false | false |
| undoFailed(15) | false | false |
| authorizationError(16) | false | true |
| notWritable(17) | false | false |
| inconsistentName(18) | false | false |

You can override this behavior by setting a property inside \${OPENNMS_HOME}/etc/opennms.properties in the form:

org.opennms.netmgt.snmp.errorStatus.[statusCode].[type]

For example, to make authorizationError(16) abort and not retry, you would set:

org.opennms.netmgt.snmp.errorStatus.16.fatal=true
org.opennms.netmgt.snmp.errorStatus.16.retry=false