Installation Guide

OpenNMS Horizon 21.0.1, Last updated 2017-11-16 11:34:40 EST

Table of Contents

Chapter 1. Installation Overview

The *OpenNMS* platform can be installed in several ways. This guide describes the installation of the platform on *Red Hat Enterprise Linux (RHEL)*-based, *Debian*-based and *Microsoft Windows* operating systems. The following abbreviations will be used to refer to the following operating systems:

- *RHEL*: Red Hat Enterprise Linux 6 or higher, CentOS 6 or higher, Fedora 20 or higher
- *Debian*: Debian 7 or higher, Ubuntu 14.04 or higher
- *Microsoft Windows*: Windows 8.1, Windows Server 2012, Windows 10

Installable, precompiled software packages are provided through *RHEL Yum* and *Debian APT* repository servers and from the OpenNMS Sourceforge project page. Installing *OpenNMS* requires the following prerequisites:

- A configured Yum or APT Package Repository for your platform (Linux only)
- Internet access to download and verify *OpenNMS* packages from the Yum or APT package repositories
- Oracle Java SE Development Kit 8 environment
- PostgreSQL database version 9.1 or higher, it has only been tested with PostgreSQL 9.1 through 9.6
- A time-series database engine to persist long-term performance data:
  - JRobin: The default choice. JRobin is included inside OpenNMS and doesn't require additional software to be installed.
  - RRDtool: A higher performance, file-based database.
  - Newts: The highest performance solution. Newts uses an Apache Cassandra database for clustered scalability.

Please make sure your DNS settings for the OpenNMS server are correct. In case there is an incorrect or missing *A Resource Record* for the server, OpenNMS might not start correctly. The reason is that the Java security manager cannot be initialized and an *RMI class loader disabled* exception will be shown.

OpenJDK 8 can be used, but for production and critical environments *Oracle Java SE Development Kit 8* is recommended.

`${OPENNMS_HOME}` will be used to refer to the path where *OpenNMS* is installed. It is different depending on your platform:

- *RHEL*: `/opt/opennms`
- *Debian*: `/usr/share/opennms`
- *Microsoft Windows*: `C:\Program Files\opennms`

With the *opennms* meta package all dependencies needed for the components mentioned above are maintained. The following sections describe how to install *OpenNMS* on a single system. Dependencies for *Java* and the *PostgreSQL* database are maintained with the *opennms* meta installation package.

Chapter 2. Compatibility Matrix

OpenNMS	Java	PostgreSQL	Newts	Cassandra	Grafana	Elasticsearch	Kafka
Horizon 19	8	9.2+	**Yes**	**1.2, 2.X, 3.0**	**3.X, 4.X**	**2.X, 5.X**	**0.10**
Horizon 18	8	9.1+	**Yes**	**1.2, 2.X, 3.0**	**3.X, 4.X**	**1.0**	No
Meridian 2016	8	9.1+	**Yes**	**1.2, 2.X, 3.0**	**3.X, 4.X**	No	No
Horizon 17	8	9.1+	**Yes**	**1.2, 2.X**	**3.X, 4.X**	No	No
Horizon 16	8	9.0+	No	No	**3.X, 4.X**	No	No
Horizon 15	7	9.0+	No	No	No	No	No
Meridian 2015	7	9.0+	No	No	No	No	No

Chapter 3. Yum/APT Package Repositories

Installation packages are available for different releases of *OpenNMS*. You need to choose which release you would like to run and then configure your package repository to point to that release. Configuring a package repository will enable you to install and update the software by using standard Linux software update tools like *yum* and *apt*.

The following package repositories are available:

Table 1. OpenNMS package repositories

Release	Description
`stable`	Latest stable release. This version is recommended for all users.
`testing`	Release candidate for the next stable release.
`snapshot`	Latest successful development build, the "nightly" build.
`branches/${BRANCH-NAME}`	Install from a specific branch name for testing a specific feature that is under development. Available branches can be found in https://yum.opennms.org/branches/ or https://debian.opennms.org/dists/branches/.

To install a different release the repository files have to be installed and manually modified.

3.1. RHEL Yum Repository

Install the configuration for a package repository

```
rpm -Uvh https://yum.opennms.org/repofiles/opennms-repo-${RELEASE}-rhel7.noarch.rpm ①
rpm --import https://yum.opennms.org/OPENNMS-GPG-KEY
```

① Replace `${RELEASE}` with a release name like `stable` (recommended), `testing`, or `snapshot`.

3.2. Debian APT Repository

Create a new apt source file (eg: `/etc/apt/sources.list.d/opennms.list`), and add the following 2 lines:

Package repository configuration for Debian-based systems

```
deb https://debian.opennms.org ${RELEASE} main ①
deb-src https://debian.opennms.org ${RELEASE} main ①
```

① Replace `${RELEASE}` with a release name like `stable` (recommended), `testing`, or `snapshot`.

Import the packages' authentication key with the following command:

GPG key import for Debian-based systems

```
wget -O - https://debian.opennms.org/OPENNMS-GPG-KEY | apt-key add -
```

Chapter 4. OpenNMS

After configuring the package repository, you are ready to install the *OpenNMS Horizon* packages, configure the database, and initialize the *OpenNMS Horizon* platform.

4.1. RHEL

This section describes how to install the *OpenNMS* platform on *CentOS 7.1*. The setup process is described in the following steps:

1. Installation of the *opennms* meta package which handles all dependencies
2. Initialize *PostgreSQL* database and configure access
3. Initialize *OpenNMS* and first start of the application

4.1.1. Install OpenNMS

Installation of the full application with all dependencies like PostgreSQL and Java

```
yum -y install opennms
```

The following packages will be automatically installed:

- *opennms*: The platform meta package which handles all dependencies from *OpenNMS* repository.
- *jicmp6* and *jicmp*: *Java* bridge to allow sending *ICMP messages* from *OpenNMS* repository.
- *opennms-core*: *OpenNMS* core services, e.g. *Provisiond*, *Pollerd* and *Collectd* from *OpenNMS* repository.
- *opennms-webapp-jetty*: *OpenNMS* web application from *OpenNMS* repository
- *jdk1.8*: *Oracle Java SE Development Kit 8* environment from *OpenNMS* respository
- *postgresql*: *PostgreSQL* database server from distribution repository
- *postgresql-libs*: *PostgreSQL* database from distribution repository

Verify the version of the *OpenNMS* packages that was installed with `yum info opennms`.

With the successful installed packages the *OpenNMS* platform is installed in the following directory structure:

```
[root@localhost /opt/opennms]# tree -L 2
.
└── opennms
    ├── bin
    ├── contrib
    ├── data
    ├── deploy
    ├── etc
    ├── jetty-webapps
    ├── lib
    ├── logs -> /var/log/opennms
    ├── share -> /var/opennms
    └── system
```

4.1.2. Disable Automatic Updates (Optional)

We recommend you disable the OpenNMS Horizon Yum repository to avoid upgrades while it is running.

OpenNMS Horizon requires some manual steps upon upgrade to make sure the database and configuration are consistent in the new version, and on systems with many nodes or lots of events, this can take hours. For this reason, it is recommended to exclude the OpenNMS Horizon packages from yum except when you are planning on performing an upgrade.

To do so, edit the `/etc/yum.repos.d/opennms-*.repo` file and change `enabled=1` to `enabled=0` in each section.

When you are ready to upgrade OpenNMS Horizon, call yum with the `--enablerepo` option to turn the 2 repositories defined in this file back on. For example, if you installed the stable repository *RPM* on a *CentOS* or *RHEL 7* system, you would run:

```
yum -y --enablerepo=opennms-repo-stable-common \
  --enablerepo=opennms-repo-stable-rhel7 \
  upgrade opennms
```

4.1.3. Prepare PostgreSQL

The *CentOS* package installs but doesn't initialize the *PostgreSQL* database directory. Additionally *OpenNMS* requires authentication to access the database and are described in this section. Initialize the database directory with

Initialization of the PostgreSQL database

```
postgresql-setup initdb
```

System startup configuration for PostgreSQL

```
systemctl enable postgresql
```

Startup PostgreSQL database

```
systemctl start postgresql
```

The next step is setting the *postgres* super user password and creating an *opennms* database user with password. Additionally it is required to configure the authentication method to allow authentication from the local network.

Accounting and database management for OpenNMS

```
su - postgres
createuser -P opennms
createdb -O opennms opennms
exit
```

Set password for Postgres super user

```
su - postgres
psql -c "ALTER USER postgres WITH PASSWORD 'YOUR-POSTGRES-PASSWORD';"
exit
```

The super user is required to be able to initialize and change the database schema for installation and updates.

To allow *OpenNMS* access to the database over the local network *PostgreSQL* has to be configured.

```
vi /var/lib/pgsql/data/pg_hba.conf
```

Configuration of network access for PostgreSQL

```
host    all             all             127.0.0.1/32            md5 ①
host    all             all             ::1/128                 md5 ①
```

① Change method from `ident` to `md5` for *IPv4* and *IPv6* on localhost.

Apply configuration changes for PostgreSQL

```
systemctl reload postgresql
```

In the next step configure the *OpenNMS* database configuration.

```
vi ${OPENNMS_HOME}/etc/opennms-datasources.xml
```

Configuration for database authentication in OpenNMS

```
<jdbc-data-source name="opennms"
                  database-name="opennms"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/opennms"
                  user-name="** YOUR-OPENNMS-USERNAME **" ①
                  password="** YOUR-OPENNMS-PASSWORD **" /> ②

<jdbc-data-source name="opennms-admin"
                  database-name="template1"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/template1"
                  user-name="postgres" ③
                  password="** YOUR-POSTGRES-PASSWORD **" /> ④
```

① Set the user name to access the *OpenNMS* database table

② Set the password to access the *OpenNMS* database table

③ Set the *postgres* user for administrative access to PostgreSQL

④ Set the password for administrative access to PostgreSQL

4.1.4. Initialize OpenNMS

OpenNMS is now configured to access the database. It is required to set the *Java* environment running *OpenNMS* and initialize the database schema.

Configuration of Java environment for OpenNMS

```
${OPENNMS_HOME}/bin/runjava -s
```

Initialization of database and system libraries

```
${OPENNMS_HOME}/bin/install -dis
```

System startup configuration for OpenNMS

```
systemctl enable opennms
```

Startup OpenNMS

```
systemctl start opennms
```

After starting *OpenNMS* the web application can be accessed on http://<ip-or-fqdn-of-your-

server>:8980/opennms. The default login user is *admin* and the password is initialized to *admin*.

Change the default admin password to a secure password immediately.

4.2. Debian

This guide does not apply to OpenNMS Meridian, which can be installed only on Red Hat Enterprise Linux or CentOS systems.

This section describes how to install the *OpenNMS* platform on *Ubuntu 14.04 LTS*. The setup process is described in the following steps:

1. Installation of the *opennms* meta package which handles all dependencies
2. Initialize *PostgreSQL* database and configure access
3. Initialize *OpenNMS* and first start of the application

4.2.1. Install OpenNMS

Installation of the full application with all dependencies like PostgreSQL and Java

```
apt-get update
apt-get install -y opennms
```

The following packages will be automatically installed:

- *opennms*: The platform meta package which handles all dependencies from *OpenNMS* repository.
- *jicmp6* and *jicmp*: *Java* bridge to allow sending *ICMP messages* from *OpenNMS* repository.
- *opennms-core*: *OpenNMS* core services, e.g. *Provisiond*, *Pollerd* and *Collectd* from *OpenNMS* repository.
- *opennms-webapp-jetty*: *OpenNMS* web application from *OpenNMS* repository
- *jdk1.8*: *Oracle Java 8* environment from *OpenNMS* respository
- *postgresql*: *PostgreSQL* database server from distribution repository
- *postgresql-libs*: *PostgreSQL* database from distribution repository

Verify the version of the *OpenNMS* packages that was installed with `apt-cache show opennms`.

With the successful installed packages the *OpenNMS* platform is installed in the following directory structure:

```
[root@localhost /usr/share/opennms]# tree -L 2
.
└── opennms
    ├── bin
    ├── data
    ├── deploy
    ├── etc -> /etc/opennms
    ├── instances
    ├── jetty-webapps
    ├── lib -> ../java/opennms
    ├── logs -> /var/log/opennms
    ├── share -> /var/lib/opennms
    └── system
```

4.2.2. Disable Automatic Updates (Optional)

We recommend you disable automatic updating of the OpenNMS Horizon packages to avoid upgrades while it is running.

OpenNMS Horizon requires some manual steps upon upgrade to make sure the database and configuration are consistent in the new version, and on systems with many nodes or lots of events, this can take hours. For this reason, it is recommended to exclude the OpenNMS Horizon packages from update except when you are planning on performing an upgrade.

To do so, run:

```
sudo apt-mark hold libopennms-java \
     libopennmsdeps-java \
     opennms-common opennms-db
```

When you are ready to upgrade OpenNMS Horizon, unhold them and upgrade, then hold them again:

```
sudo apt-mark unhold libopennms-java \
     libopennmsdeps-java \
     opennms-common opennms-db

sudo apt-get upgrade opennms

sudo apt-mark hold libopennms-java \
     libopennmsdeps-java \
     opennms-common opennms-db
```

4.2.3. Prepare PostgreSQL

The *Debian* package installs also *PostgreSQL* database and is already initialized and added in the

runlevel configuration. It is only necessary to start the *PostgreSQL* database without a restart.

Startup PostgreSQL database

```
service postgresql start
```

The next step is creating an *opennms* database user with password and configure the authentication method.

Accounting and database management for OpenNMS

```
su - postgres
createuser -P opennms
createdb -O opennms opennms
exit
```

It is not necessary to change the authentication method in `pg_hba.conf`, it is by default set to `md5` for localhost connections.

Set password for Postgres super user

```
su - postgres
psql -c "ALTER USER postgres WITH PASSWORD 'YOUR-POSTGRES-PASSWORD';"
exit
```

The super user is required to be able to initialize and change the database schema for installation and updates.

```
vi ${OPENNMS_HOME}/etc/opennms-datasources.xml
```

Configuration for database authentication in OpenNMS

```
<jdbc-data-source name="opennms"
                  database-name="opennms"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/opennms"
                  user-name="** YOUR-OPENNMS-USERNAME **" ①
                  password="** YOUR-OPENNMS-PASSWORD **" /> ②

<jdbc-data-source name="opennms-admin"
                  database-name="template1"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/template1"
                  user-name="postgres" ③
                  password="** YOUR-POSTGRES-PASSWORD **" /> ④
```

① Set the user name to access the *OpenNMS* database table

② Set the password to access the *OpenNMS* database table

③ Set the *postgres* user for administrative access to PostgreSQL

④ Set the password for administrative access to PostgreSQL

4.2.4. Initialize OpenNMS

OpenNMS is now configured to access the database. It is required to set the *Java* environment running *OpenNMS* and initialize the database schema.

Configuration of Java environment for OpenNMS

```
${OPENNMS_HOME}/bin/runjava -s
```

Initialization of database and system libraries

```
${OPENNMS_HOME}/bin/install -dis
```

It is not necessary to add *OpenNMS* to the run level manually, it is automatically added after setup.

Startup OpenNMS

```
service opennms start
```

After starting OpenNMS, the web application can be accessed on http://<ip-or-fqdn-of-your-server>:8980/opennms. The default login user is *admin* and the password is initialized to *admin*.

Change the default admin password to a secure password immediately.

4.3. Microsoft Windows

This guide does not apply to OpenNMS Meridian, which can be installed only on Red Hat Enterprise Linux or CentOS systems.

OpenNMS is mostly developed on Unix/Linux based systems, nevertheless it is possible to install the platform on *Microsoft Windows* operating systems. To install the application a graphical installer is provided and can be used to install *OpenNMS* on *Microsoft Windows*. This section describes how to install the *OpenNMS* platform on *Microsoft Windows 2012 Server*.

The standalone installer for *Microsoft Windows* is only available for the most recent stable version of *OpenNMS*.

It is required to have Oracle JDK 8 installed. The *JRE* is **NOT** sufficient.

To edit *OpenNMS* configuration files on *Microsoft Windows* the tool Notepad++ can deal with the formatting of *.property* and *.xml* files.

The setup process is described in the following steps:

1. Installation of *PostgreSQL* database service
2. Download and install the graphical *OpenNMS* installer
3. First start of the *OpenNMS* application

4.3.1. Install PostgreSQL

PostgreSQL is available for *Microsoft Windows* and latest version can be downloaded from Download PostgreSQL page. Follow the on-screen instructions of the graphical installer.

The placeholder `{PG-VERSION}` represents the *PostgreSQL* version number. A version of *9.1+* is required for *OpenNMS*.

The following information has to be provided:

- Installation directory for *PostgreSQL*, e.g. `C:\Program Files\PostgreSQL{PG-VERSION}`
- Password for the database superuser (*postgres*), this password will be used during the *OpenNMS* setup.
- Port to listen for *PostgreSQL* connections, default is `5432` and can normally be used.
- Locale for the database, keep `[Default locale]`, if you change the locale, *OpenNMS* may not be able to initialize the database.

It is not required to install anything additional from the *PostgreSQL Stack Builder*.

The database data directory is automatically initialized during the setup and the *postgresql-x64-{PG-VERSION}* is already added as service and automatically started at system boot.

It is not necessary to change the authentication method in `pg_hba.conf`, it is by default set to `md5` for localhost connections.

4.3.2. Install OpenNMS with GUI installer

For *Microsoft Windows* environments download the *standalone-opennms-installer-{ONMS-VERSION}.zip* file from the OpenNMS SourceForge repository. Extract the downloaded *ZIP* file.

The `{ONMS-VERSION}` has to be replaced with the latest stable version.

Start the graphical installer and follow the on screen instructions. The following information has to be provided:

- Path to *Oracle JDK*, e.g. `C:\Program Files\Java\jdk1.8.0_51`

- Installation path for *OpenNMS*, e.g. `C:\Program Files\OpenNMS`
- Select packages which has to be installed, the minimum default selection is *Core* and *Docs*
- PostgreSQL Database connection
  - Host: Server with *PostgreSQL* running, e.g. `localhost`
  - Name: Database name for *OpenNMS*, e.g. `opennms`
  - Port: *TCP* port connecting to *PostgreSQL* server, e.g. `5432`
  - Username (administrative superuser): *PostgreSQL* superuser, e.g. `postgres`
  - Password (administrative superuser): Password given during *PostgreSQL* setup for the superuser
  - Username (runtime user for opennms): Username to connect to the *OpenNMS* database, e.g. `opennms`
  - Password (runtime user for opennms): Password to connect to the *OpenNMS* database, e.g. `opennms`
- Configure a discovery range for an initial node discovery. If you don't want any discovery set begin and end to the same unreachable address.

Choose secure passwords for all database users and don't use the example passwords above in production.

There is currently an open issue in the installer NMS-7831. Username and password are not written to the `opennms-datasources.xml` file and has to be changed manually. The initialize of the database will fail with an authentication error.

Configuration for database authentication in OpenNMS

```
<jdbc-data-source name="opennms"
                  database-name="opennms"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/opennms"
                  user-name="** YOUR-OPENNMS-USERNAME **" ①
                  password="** YOUR-OPENNMS-PASSWORD **" /> ②

<jdbc-data-source name="opennms-admin"
                  database-name="template1"
                  class-name="org.postgresql.Driver"
                  url="jdbc:postgresql://localhost:5432/template1"
                  user-name="postgres" ③
                  password="** YOUR-POSTGRES-PASSWORD **" /> ④
```

① Set the user name to access the *OpenNMS* database table

② Set the password to access the *OpenNMS* database table

③ Set the *postgres* user for administrative access to PostgreSQL

④ Set the password for administrative changes of the *OpenNMS* database table

After setting the username and passwords in `opennms-datasources.xml` re-run the graphical installer and also initialize the database. *OpenNMS* can be started and stopped with the `start.bat` and `stop.bat` script located in `%OPENNMS_HOME%\bin` directory.

After starting *OpenNMS* with the `start.bat` file the web application can be accessed on http://<ip-or-fqdn-of-your-server>:8980/opennms. The default login user is *admin* and the password is initialized to *admin*.

Change the default admin password to a secure password immediately.

The Wiki article Configuring OpenNMS as Windows Service describes how to create a *Windows Service* from the `start.bat` files. There is also a Java Wrapper which allows to install *Java* applications as *Windows Service*.

Chapter 5. Oracle Java SE Development Kit 8

Installing the *Oracle Java SE Development Kit 8* (*JDK8*) requires installation packages provided by *Oracle* or a 3rd-party maintainer for *Debian*-based Linux distributions. The following tools should be installed to follow this installation manual:

- Download files and tools with `wget` and `curl`
- Extract archives with `tar`
- Text manipulation with `sed`
- Editing text, e.g. `vi`, `nano` or `joe`
- Internet access

By downloading the *Oracle Java SE Development Kit 8* RPM installer, you will accept the license agreement from *Oracle* which can be found on the Java distribution web site.

Installing the *Java Runtime Environment (JRE)* is not sufficient. The development kit is often named *openjdk-devel* or *openjdk-jdk*. With a *JRE* installed, *OpenNMS Horizon* will not start and throws a `java.lang.ClassNotFoundException: com.sun.tools.attach.AttachNotSupportedException`. For more details see NMS-9327.

5.1. RHEL

This section describes how to install *Oracle Java SE Development Kit 8* on a *RPM-based* system like *Red Hat Enterprise Linux 7* or *CentOS 7.1*.

Download Oracle JDK RPM

```
wget --no-cookies \
     --no-check-certificate \
     --header \
              "Cookie: oraclelicense=accept-securebackup-cookie" \
              "http://download.oracle.com/otn-pub/java/jdk/8u45-b14/jdk-8u45-linux-
x64.rpm" \
     -O /tmp/jdk-8-linux-x64.rpm
```

Install Oracle JDK RPM file

```
yum install /tmp/jdk-8-linux-x64.rpm
```

5.2. Debian

This section describes how to install *Oracle Java SE Development Kit 8* on a *Debian-based* system like *Debian 8* or *Ubuntu 14.04 LTS*.

Add Java repository from webupd8 maintainer

```
su -
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee
/etc/apt/sources.list.d/webupd8team-java.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a
/etc/apt/sources.list.d/webupd8team-java.list
```

Add repository key server and update repository

```
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
apt-get update
```

Install Oracle Java SE Development Kit 8

```
apt-get install -y oracle-java8-installer
```

5.3. Microsoft Windows

This section describes how to install *Oracle Java SE Development Kit 8* on a system running the *Microsoft Windows Server 2012* operating system.

Download the Microsoft Windows Java SE Development Kit 8 installer with PowerShell or a browser

```
cd C:\Users\Administrator\Downloads
Invoke-WebRequest http://javadl.sun.com/webapps/download/AutoDL?BundleId=107944
-Outfile java8-installer.exe
```

Start the `java8-installer.exe` from the command line or with *Windows Explorer* from the Administrator's *Download* folder.

The setup requires administrative privileges.

5.4. Java Environment

To locate the *Java* system files, applications typically use the `$JAVA_HOME` environment variable. The environment can be set for a specific user or globally for the whole system on boot time.

Example path to Java on RHEL, Debian and Microsoft Windows systems

- RHEL: `/usr/java/jdk1.8.0_51`

- Debian: `/usr/lib/jvm/java-8-oracle`
- Microsoft Windows: `C:\Program Files\Java\jre1.8.0_51`

5.4.1. Set JAVA_HOME on Linux

Option 1: Set the Java environment for the current user

```
vi ~/.bash_profile
export JAVA_HOME=/path/to/java
```

Option 2: Set the Java environment for all users on boot time

```
vi /etc/profile
export JAVA_HOME=/path/to/java
```

5.4.2. Set JAVA_HOME on Microsoft Windows

Option 1: Set JAVA_HOME as user specific system variable

```
setx "JAVA_HOME" "path\to\java"
```

Option 2: Set JAVA_HOME as a System variable

```
setx /M "JAVA_HOME" "path\to\java"
```

Chapter 6. RRDtool

In most *Open Source* applications, RRDtool is often used and is the de-facto open standard for *Time Series Data*. The basic installation of *OpenNMS* comes with *JRobin* but it is simple to switch the system to use *RRDtool* to persist *Time Series Data*. This section describes how to install *RRDtool*, the *jrrd2 OpenNMS Java Interface* and how to configure *OpenNMS* to use it. *RRDtool* can be installed from the official package repositories provided by *RHEL* and *Debian* based *Linux* distributions.

6.1. RHEL

Installation on RHEL/CentOS

```
yum install rrdtool
```

6.2. Debian

Installation of RRDtool on Debian/Ubuntu

```
apt-get install rrdtool
```

6.3. Source

If you want the latest version of RRDtool, you may want to compile it from source. Instructions for doing so are at rrdbuild.

The latest version of RRDtool may not always be compatible with the version of *OpenNMS* that you want to run. Please ask about RRDtool support on the discussion lists or chat rooms if you have any problems running a new version of RRDtool.

If you want to install the latest *RRDtool* from source, make sure the `rrdtool` binary is in search path. To make the setup easier, you can link the binary to `/usr/bin/rrdtool` which is the location where *OpenNMS* will expect to find the executable binary.

6.4. Install jrrd2 Interface

To get access from the *OpenNMS Java Virtual Machine* you have to install *jrrd2* as an interface. You can install it from the *OpenNMS* package repository with:

Installation of jrrd2 on RHEL/CentOS

```
yum install jrrd2
```

Installation of jrrd2 on Debian/Ubuntu

```
apt-get install jrrd2
```

With OpenNMS 17.0.0 it is preferred to use *jrrd2* instead of *jrrd*. The *jrrd2* module is improved for performance by adding multithreading capabilities.

6.5. Configure OpenNMS Horizon

To configure *OpenNMS* to use *RRDtool* instead of *JRobin* configure the following properties in `rrd-configuration.properties`.

Configuration of RRDtool in OpenNMS on RHEL/CentOS

```
org.opennms.rrd.strategyClass=org.opennms.netmgt.rrd.rrdtool.MultithreadedJniRrdStrate
gy
org.opennms.rrd.interfaceJar=/usr/share/java/jrrd2.jar
opennms.library.jrrd2=/usr/lib64/libjrrd2.so
```

Configuration of RRDtool in OpenNMS on Debian/Ubuntu

```
org.opennms.rrd.strategyClass=org.opennms.netmgt.rrd.rrdtool.MultithreadedJniRrdStrate
gy
org.opennms.rrd.interfaceJar=/usr/share/java/jrrd2.jar
opennms.library.jrrd2=/usr/lib/jni/libjrrd2.so
```

OpenNMS expects the *RRDtool* binary in `/usr/bin/rrdtool`.

Table 2. References to the RRDtool binary

Configuration file	Property
`opennms.properties`	`rrd.binary=/usr/bin/rrdtool`
`response-adhoc-graph.properties`	`command.prefix=/usr/bin/rrdtool`
`response-graph.properties`	`command.prefix=/usr/bin/rrdtool` `info.command=/usr/bin/rrdtool`
`snmp-adhoc-graph.properties`	`command.prefix=/usr/bin/rrdtool`
`snmp-graph.properties`	`command.prefix=/usr/bin/rrdtool` `command=/usr/bin/rrdtool info`

Chapter 7. Newts

Newts is a time-series data store based on Apache Cassandra. *Newts* is a persistence strategy, that can be used as an alternative to JRobin or RRDtool.

It is currently not supported to initialize the *Newts* keyspace from *Microsoft Windows Server* operating system. *Microsoft Windows* based *Cassandra* server can be part of the cluster, but keyspace initialization is only possible using a _Linux-_based system.

7.1. Setting up Cassandra

Cassandra is only required when using *Newts*. If your *OpenNMS Horizon* system is not using *Newts*, you can skip this section.

It is recommended to install *Cassandra* on a dedicated server, but is also possible to run a node on the *OpenNMS Horizon* server itself. This installation guide describes how to set up a single *Cassandra* instance on the same system as *OpenNMS Horizon* for the purpose of evaluating and testing *Newts*. These steps are not suitable for a production *Cassandra Cluster*. If you already have a running cluster you can skip this section.

For further information see Cassandra Getting Started Guide. Before setting up a production cluster make sure to consult Anti-patterns in Cassandra.

7.1.1. RHEL

This section describes how to install the latest *Cassandra 3.0.x* release on a *RHEL* based systems for *Newts*. The first step is to add the *DataStax* community repository and install the required *GPG Key* to verify the integrity of the *RPM packages*. After that install the package with *yum* and the *Cassandra* service is managed by *Systemd*.

This description was built on *CentOS 7.2*.

Cassandra 3.x requires Java 8+. See installing Java on RHEL for instructions.

Add the DataStax repository

```
vi /etc/yum.repos.d/datastax.repo
```

Content of the datastax.repo file

```
[datastax]
name = "DataStax Repo for Apache Cassandra"
baseurl = https://rpm.datastax.com/community
enabled = 1
gpgcheck = 1
```

Install GPG key to verify RPM packages

```
rpm --import https://rpm.datastax.com/rpm/repo_key
```

Install latest Cassandra 3.0.x package

```
yum install dsc30
```

Enable Cassandra to start on system boot

```
chkconfig cassandra on
```

Start cassandra service

```
service cassandra start
```

Verify whether the *Cassandra* service is automatically started after rebooting the server.

7.1.2. Debian

This section describes how to install the latest *Cassandra 3.0.x* release on a *Debian*-based system for *Newts*. The first step is to add the *DataStax* community repository and install the required *GPG Key* to verify the integrity of the *DEB packages*. After that install the packages with *apt* and the *Cassandra* service is added to the runlevel configuration.

This description was built on *Debian 8.3* and *Ubuntu 16.04 LTS*.

Cassandra 3.x requires Java 8+. See installing Java on Debian for instructions.

Add the DataStax repository

```
vi /etc/apt/sources.list.d/cassandra.sources.list
```

Content of the cassandra.sources.list file

```
deb https://debian.datastax.com/community stable main
```

Install GPG key to verify DEB packages

```
wget -O - https://debian.datastax.com/debian/repo_key | apt-key add -
```

Install latest Cassandra 3.0.x package

```
apt-get update
apt-get install dsc30
```

The *Cassandra* service is added to the runlevel configuration and is automatically started after installing the package.

Verify whether the *Cassandra* service is automatically started after rebooting the server.

7.1.3. Microsoft Windows

This section describes how to install the latest *Cassandra 3.0.x* release on a *Microsoft Windows Server* based systems for *Newts*. The first step is to download the graphical installer and register *Cassandra* as a *Windows Service* so it can be manged through the *Service Manager*.

This description was built on *Windows Server 2012*.

Cassandra 3.x requires Java 8+. See installing Java on Windows for instructions.

Download the DataStax graphical installer for Cassandra from PowerShell or a Browser

```
cd C:\Users\Administrator\Downloads
Invoke-WebRequest https://downloads.datastax.com/community/datastax-community-
64bit_3.0.6.msi -Outfile datastax-community-64bit_3.0.6.msi
```

Run the Windows Installer file from *PowerShell* or through *Windows Explorer* and follow the setup wizard to install. During the installation, accept the options to automatically start the services. By default the *DataStax Server*, *OpsCenter Server* and the *OpsCenter Agent* will be automatically installed and started.

The *DataStax OpsCenter Server* is only required to be installed once per *Cassandra Cluster*.

If you install the *DataStax OpsCenter* make sure you have *Chrome* or *Firefox* installed.

7.2. Configure OpenNMS Horizon

Once *Cassandra* is installed, *OpenNMS Horizon* can be configured to use *Newts*. To enable and configure *Newts*, set the following properties in `${OPENNMS_HOME}/etc/opennms.properties`:

Configuration for OpenNMS Horizon

```
# Configure storage strategy
org.opennms.rrd.storeByForeignSource=true
org.opennms.timeseries.strategy=newts

# Configure Newts time series storage connection
org.opennms.newts.config.hostname=$ipaddress$
org.opennms.newts.config.keyspace=newts
org.opennms.newts.config.port=9042
```

The `org.opennms.newts.config.hostname` property also accepts a comma separated list of hostnames and or IP addresses.

Once *Newts* has been enabled, you can initialize the *Newts* schema in *Cassandra* with the following:

Initialize Newts keyspace in Cassandra

```
${OPENNMS_HOME}/bin/newts init
```

Optionally, you can now connect to your *Cassandra* cluster and verify that the keyspace has been properly initialized:

Verify if the keyspace is initialized with cqlsh

```
cqlsh
use newts;
describe table terms;
describe table samples;
```

Restart *OpenNMS Horizon* to apply the changes.

Chapter 8. R Statistics System

R is a free software environment for statistical computing and graphics. *OpenNMS* can leverage the power of *R* for forecasting and advanced calculations on collected time series data.

OpenNMS interfaces with *R* via *stdin* and *stdout*, and for this reason, *R* must be installed on the same host as *OpenNMS*. Note that installing *R* is optional, and not required by any of the core components.

The *R* integration is not currently supported on *Microsoft Windows* systems.

8.1. RHEL

This section describes how to install *R* on a *RHEL* based system.

This description was built on *RHEL 7* and *CentOS 7.1*.

Install the EPEL repositories

```
yum install epel-release
```

Install R

```
yum install R
```

8.2. Debian

This section describes how to install *R* on a *Debian*-based system.

This description was built on *Debian 8* and *Ubuntu 14.04 LTS*.

Install R

```
sudo apt-get install r-recommended
```

Chapter 9. Minion

Minion gives the ability to monitor devices and applications which are in isolated networks and hard to reach from a central *OpenNMS Horizon* instance. Maintaining a large set of *Firewall* rules to allow a variety of management protocols is sometimes tedious and hard to set up. Communicating with managed devices over unreliable networks and the use of *UDP* based management protocols can also be difficult to maintain. Deploying a *Minion* can be used to address these issues.

A *Minion* can be used when a central *OpenNMS Horizon* can't reach all devices and *Management Agents* for monitoring. Furthermore it simplifies the network communication by using *TCP*-based *ActiveMQ* and *ReST* communication.

The network area where access to managed network devices and applications is allowed can be modeled in a *Location*. Monitored *Nodes* and *IP Services* are associated to *Locations* and are defined during *Provisioning*. Each *Minion* is configured with a *Location* and all *Nodes* and *IP Services* in the same *Location* are monitored through this *Minion*.

The *Minion* is currently not designed to be a replacement for the *Remote Poller*. By using the *Remote Poller* a service can be tested from several remote sites, whereas a *Minion* extends network reachability for a central *OpenNMS Horizon* instance.

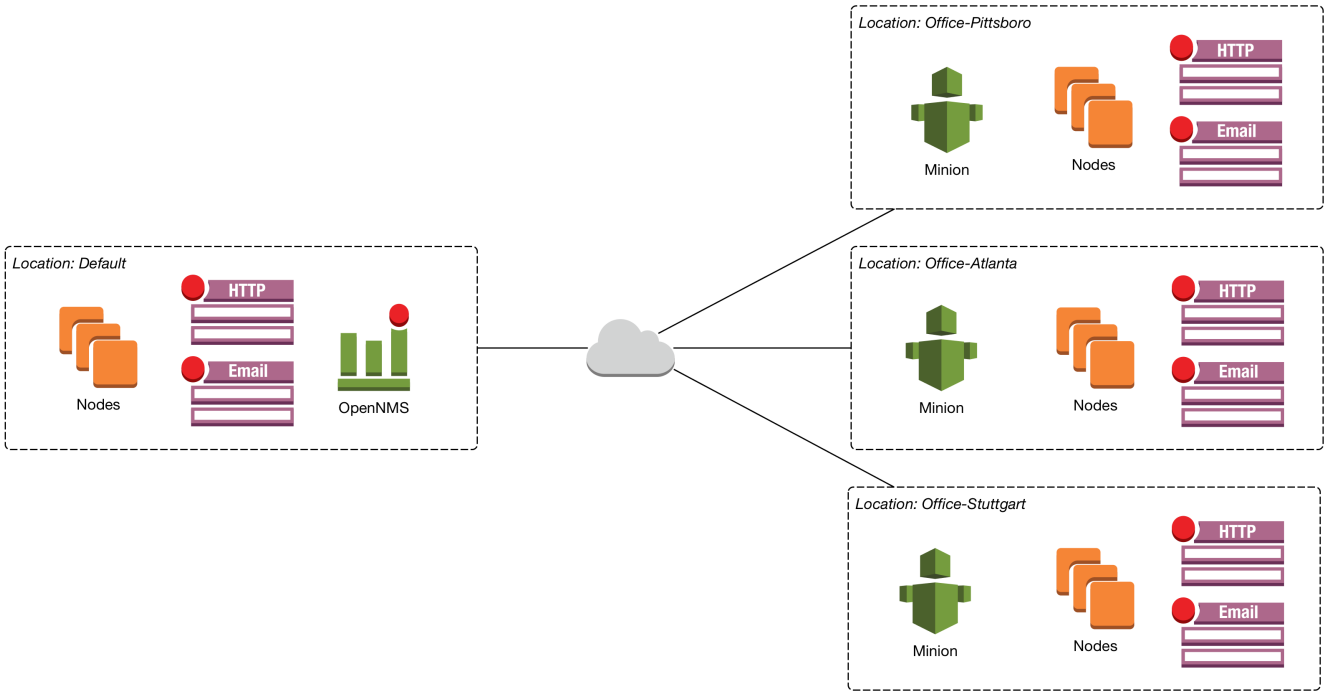

Figure 1. Nodes with Minions in Locations

The figure *Nodes* with *Minions* in *Locations* illustrates a *Minion* deployment in isolated branch offices.

Every *Node* created in *OpenNMS Horizon* is by default created in the *Location* named *Default*. All *Nodes* and *Services* in the *Default Location* are handled by the central *OpenNMS Horizon* instance itself. For each branch office in an isolated network, a *Location* is defined. The *Minion* has a configuration property for the *Location* and will register itself to the OpenNMS Horizon instance on

startup.

The *Provisioning System* allows to associate *Nodes* to a *Location*. *OpenNMS Horizon* will delegate monitoring requests for *Nodes* in the specified *Locations* to the registered *Minions* and uses them as a proxy.

Figure Minion communication gives a more detailed overview about the communication between an *OpenNMS Horizon* instance and a *Minion*.

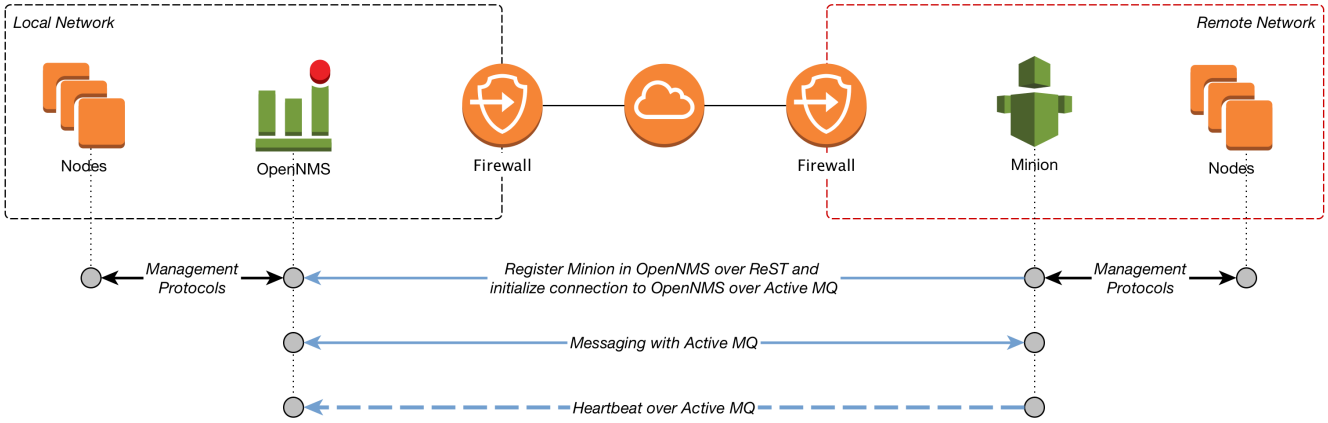

Figure 2. Minion communication

The *Minion* needs a configuration which requires at minimum the following information:

- An unique identifier (`id`) for this specific *Minion*
- Monitoring *Location* name (`location`) this *Minion* is responsible
- The communication endpoints (`broker-url` and `http-url`) for the central *OpenNMS Horizon* instance

The configuration resides in a property file in `${MINION_HOME}/etc/org.opennms.minion.controller.cfg`. When the minimal configuration is set up the *Minion* can be started and initially connects to the central *OpenNMS Horizon* instance and identifies itself with his unique *ID*.

The unique *ID* is generated when the packages get installed `/usr/bin/uuidgen -t` and is used if no *ID* is set manually. On upgrade the *ID* is not updated.

By default the *Minion* will be automatically provisioned as a *Node* in the *OpenNMS Horizon* instance and get automatically monitored with the *Minion-Heartbeat* service. The *Minion* sends heart beat messages to ensure it is running and functioning properly in this network area.

The specific management protocol messages, e.g. *SNMP*, *ICMP*, are piped through an *ActiveMQ* messaging communication channel and are executed by a *Minion*. Responses are forwarded to the central *OpenNMS Horizon* instance and are processed accordingly.

Minions can be installed on every system that is able to communicate with these two endpoints:

- The *OpenNMS ReST Interface*, by default *TCP* port 8980
- The *ActiveMQ* broker used by *OpenNMS Horizon*, by default *TCP* port 61616

The following management protocols are currently supported in a *Minion* proxy scenario:

- Receive *Syslog* messages and forward them through *ActiveMQ* to a central *OpenNMS Horizon* instance
- Receive *SNMP Traps* and forward them through *ActiveMQ* to a central *OpenNMS Horizon* instance
- Act as a proxy for *SNMP* performance data collections
- Act as a proxy for *Service Monitors* to test availability and measure response times from applications

Packages are only available for *RHEL*-based systems (*RPMS*).

To avoid issues, make sure the *Minion* and the instance of *OpenNMS Horizon* have the same version.

9.1. Installing Minion

This section describes how to install the *Minion* and how to configure it to communicate with a central *OpenNMS Horizon* instance.

Installing a distributed *OpenNMS Horizon* requires:

- Instance of *OpenNMS Horizon* needs to be same version as *Minion* Packages
- Packages are available as *RPMs* for *RHEL*-based systems and *DEBs* for *Debian*-based systems (including Ubuntu)
- *OpenNMS Horizon* needs to be installed and the communication to the *ReST* and *ActiveMQ* endpoints is possible

9.1.1. RHEL

This description was built on *RHEL 7* and *CentOS 7.1*.

Start by setting up the OpenNMS Yum repository and installing Java.

Once the *Yum* repository has been configured:

Install the Minion packages

```
yum -y install opennms-minion
```

The following packages will be automatically installed:

- *opennms-minion*: The Minion meta package
- *opennms-minion-container*: The *Karaf* OSGi container with *Minion* branding and additional management extensions

- *opennms-minion-features-core*: Core utilities and services required by the *Minion* features
- *opennms-minion-features-default*: Service-specific features

The *Minion* packages setup the following directory structure:

```
[root@localhost /opt/minion]# $ tree -L 1
.
├──── bin
├──── deploy
├──── etc
├──── lib
├──── repositories
└──── system
```

Configuring Startup

The Minion's startup configuration can be changed by editing the `/etc/sysconfig/minion` file. It allows you to override many of the defaults used at startup including the location of your JDK, how much memory to use, and what user to run as.

Starting the Minion

After successful installation a `minion` service can be started and enabled using *systemd* commands.

System startup configuration for Minion

```
systemctl enable minion
```

Startup Minion

```
systemctl start minion
```

After starting *Minion* the shell can be accessed locally on ssh://localhost:8201. The default login user is *admin* and the password is initialized to *admin*.

```
[root@localhost /root]# $ ssh -p 8201 admin@localhost
```

9.1.2. Debian/Ubuntu

Start by setting up the OpenNMS Apt repository and installing Java.

Once the *Apt* repository has been configured:

Install the Minion packages

```
apt-get update
apt-get -y install opennms-minion
```

The following packages will be automatically installed:

- *opennms-minion*: The Minion meta package
- *opennms-minion-container*: The *Karaf* OSGi container with *Minion* branding and additional management extensions
- *opennms-minion-features-core*: Core utilities and services required by the *Minion* features
- *opennms-minion-features-default*: Service-specific features

The *Minion* packages setup the following directory structure:

```
[root@localhost /usr/share/minion]# $ tree -L 1
.
├──── bin
├──── deploy
├──── etc
├──── lib
├──── repositories
└──── system
```

Additionally, symbolic links are set up pointing to `/etc/minion` and `/var/log/minion` to match Debian's expected filesystem layout.

Configuring Startup

The Minion's startup configuration can be changed by editing the `/etc/default/minion` file. It allows you to override many of the defaults used at startup including the location of your JDK, how much memory to use, and what user to run as.

Starting the Minion

After successful installation a `minion` service can be started and enabled using standard Debian commands.

System startup configuration for Minion

```
update-rc.d minion enable
```

Startup Minion

```
service minion start
```

After starting *Minion* the shell can be accessed locally on ssh://localhost:8201. The default login user is *admin* and the password is initialized to *admin*.

```
[root@localhost /root]# $ ssh -p 8201 admin@localhost
```

9.2. Configuring OpenNMS

Minions communicate with *OpenNMS Horizon* via *ReST* endpoints and via an *ActiveMQ* broker. Some configuration is required to setup and secure these communication channels.

9.2.1. Authentication and Authorization

The `minion` role includes the minimal amount of permissions required for a *Minion* to operate.

This guide will assume you have created a user called `minion`, with a password of `minion` that has been associated to the *ROLE_MINION* role.

9.2.2. Configure ActiveMQ

OpenNMS Horizon embeds an *ActiveMQ* broker which, by default, cannot be accessed remotely via the network. In order to make the *ActiveMQ* broker accessible remotely, you must edit `$OPENNMS_HOME/etc/opennms-activemq.xml` and uncomment the `transportConnector` with the `tcp://0.0.0.0:61616` URI.

```
<!-- Uncomment this line to allow external TCP connections -->
<!--
  WARNING: Access to port 61616 should be firewalled to prevent unauthorized injection
  of data into OpenNMS when this port is open.
-->
<transportConnector name="openwire" uri="tcp://0.0.0.0:61616?useJmx=false
&amp;maximumConnections=1000&amp;wireformat.maxFrameSize=104857600"/>
```

If you wish to restrict *ActiveMQ* connections to only one particular external IP address, you can change `0.0.0.0` to the prefered IP address.

9.3. Configuring Minion

This section describes how to configure *Minion* once it has been installed and started.

Once the *Minion* service is started and the *Karaf* shell is accessible, you can configure the *Minion* to point it at your *OpenNMS Horizon* instance.

By default the *Minion* is configured to communicate with *OpenNMS Horizon* via `localhost`.

Configure the Minion's location and endpoint URLs for communication with OpenNMS

```
[root@localhost /root]# $ ssh -p 8201 admin@localhost
...
admin@minion()> config:edit org.opennms.minion.controller
admin@minion()> config:property-set http-url http://opennms-fqdn:8980/opennms
admin@minion()> config:property-set broker-url failover:tcp://opennms-fqdn:61616
admin@minion()> config:property-set location Office-Pittsboro
admin@minion()> config:update
```

Include the `failover:` portion of the broker URL to allow the *Minion* to re-establish connectivity on failure. For a reference on the different URL formats, see ActiveMQ URI Protocols.

Configure the credentials to use when communicating with OpenNMS

```
admin@minion()> scv:set opennms.http minion minion
admin@minion()> scv:set opennms.broker minion minion
```

Another way to configure credentials is to use the `scvcli` utility in your *Minion* `bin` directory.

Example of configuring credentials with the command line utility `scvcli` on a RPM-based installation

```
[root@localhost /root]# $ cd /opt/minion
[root@localhost /opt/minion]# $ ./bin/scvcli set opennms.http minion minion
[root@localhost /opt/minion]# $ ./bin/scvcli set opennms.broker minion minion
```

Example of configuring credentials with the command line utility `scvcli` on a Debian-based installation

```
[root@localhost /root]# $ cd /usr/share/minion
[root@localhost /usr/share/minion]# $ ./bin/scvcli set opennms.http minion minion
[root@localhost /usr/share/minion]# $ ./bin/scvcli set opennms.broker minion minion
```

Restart the Minion after updating the credentials

```
[root@localhost /root]# $ systemctl restart minion
```

The credentials are configured separately since they are encrypted on disk.

9.4. Advanced Minion Configuration

This section provides information for advanced configuration topics, such as *Syslog*, *SNMP Trap* receiving or running in non-root environments.

9.4.1. Configure Linux to Allow Non-Root ICMP

By default, *Linux* does not allow regular users to perform `ping` operations from arbitrary programs (including *Java*). To enable the *Minion* to ping properly, you must set a `sysctl` option.

Enable User Ping (Running System)d

```
# run this command as root to allow ping by any user (does not survive reboots)
sysctl net.ipv4.ping_group_range='0 429496729'
```

If you wish to restrict the range further, use the *GID* for the user the *Minion* will run as, rather than `429496729`.

To enable this permanently, create a file in `/etc/sysctl.d/` to set the range:

/etc/sysctl.d/99-zzz-non-root-icmp.conf

```
# we start this filename with "99-zzz-" to make sure it's last, after anything else
that might have set it
net.ipv4.ping_group_range=0 429496729
```

9.4.2. Configure Minion to Receive Traps

If you wish your Minion to listen to *SNMP Traps*, you will need to configure your firewall to port forward from the privileged trap port (162) to the Minion's default trap listener on port 1162.

Forward 162 to 1162 with Firewalld

```
# enable masquerade to allow port-forwards
firewall-cmd --add-masquerade
# forward port 162 TCP and UDP to port 1162 on localhost
firewall-cmd --add-forward-port=port=162:proto=udp:toport=1162:toaddr=127.0.0.1
firewall-cmd --add-forward-port=port=162:proto=tcp:toport=1162:toaddr=127.0.0.1
```

9.4.3. Configure Minion to Receive Syslog Messages

If you wish your Minion to listen to syslog messages, you will need to configure your firewall to port forward from the privileged *Syslog* port (514) to the Minion's default syslog listener on port 1514.

Forward 514 to 1514 with Firewalld

```
# enable masquerade to allow port-forwards
firewall-cmd --add-masquerade
# forward port 514 TCP and UDP to port 1514 on localhost
firewall-cmd --add-forward-port=port=514:proto=udp:toport=1514:toaddr=127.0.0.1
firewall-cmd --add-forward-port=port=514:proto=tcp:toport=1514:toaddr=127.0.0.1
```

9.4.4. Minion Configuration File

Beside manually configuring a *Minion* instance via the *Karaf CLI* it is possibleto modify and deploy its configuration file through configuration management tools. The configuration file is located in `${MINION_HOME}/etc/org.opennms.minion.controller.cfg`. All configurations set in *Karaf CLI* will be persisted in this configuration file which can also be populated through configuration management tools.

Configuration file for Minion

```
id = 00000000-0000-0000-0000-deadbeef0001
location = MINION
broker-url = tcp://myopennms.example.org:61616
http-url = http://myopennms.example.org:8980/opennms
```

The *Minion* needs to be restarted when this configuration file is changed.

In case the credentials needs to be set through the *CLI* with configuration management tools or scripts, the `/opt/minion/bin/client` command can be used which allows to execute *Karaf* commands through the Linux shell.

9.5. Troubleshooting

This section gives some hints how to troubleshoot *Minion* deployments. The *OpenNMS Horizon* comes with a few useful built-in commands to verify configurations and debug access to *Management Agents* through *Minion*.

9.5.1. Verifying Connectivity

Once the URLs and credentials for communicating with the *OpenNMS Horizon* instance are configured, you can verify connectivity using:

Verify connectivity with the OpenNMS Horizon endpoints

```
admin@minion()> minion:ping
Connecting to ReST...
OK
Connecting to Broker...
OK
admin@minion()>
```

9.5.2. Execute SNMP commands through a *Minion*

The commands are available in the *Karaf CLI* on the *OpenNMS Horizon* system.

Run SNMP commands from OpenNMS Horizon in a monitoring location

```
ssh -oHostKeyAlgorithms=+ssh-dss -p 8101 localhost
snmp:walk -l MyLocation IpAddressInMyLocation 1.3.6.1.4.1
```

This commmand allows you to verify the connection and *SNMP community* configuration for a given host in a remote location.

9.5.3. Run a monitor through a *Minion*

Show all available monitors

```
poller:list-monitors
```

Run an ICMP monitor through a Minion

```
poller:poll -l MyLocation -t Time-To-Live-in-ms
org.opennms.netmgt.poller.monitors.IcmpMonitor myIpAddress
```

The *Time To Live (TTL)* is only related to messages in the *ActiveMQ* communication. In case a poll is triggered manually through *Karaf CLI* the message *TTL* in *ActiveMQ* should be at least the number of retries x timeout in ms, e.g. 3 x 2000ms = 6000ms. By default the configured polling interval is used, which is by default 5 minutes (300000 ms).