RoseNMS Documentation

Release 0.1

Craig Small

April 02, 2016

Contents

Contents:

CHAPTER 1

Introduction

RoseNMS (rnms) is, as the name implies, a Network Management System. What this means is rnms is a piece of software that gathers information on devices out on a network and tries to meaningfully interpret them to make monitoring and managment simpler.

rnms is written in python and is based upon the Turbogears 2 web framework. The basic concept is largely built around the ideas that were put into another NMS program called JFFNMS.

For more updates, please visit the RoseNMS website

CHAPTER 2

History of RoseNMS

Rnms is the third network management system that I have worked on. In the early 2000's there was a design which was not much more than some penciled scribbles for something along the lines of logcheck. That program was called GEMS (Generic Event Management System) and didn't progress past the concept stage.

What accelerated GEMS' demise was a project called Just For Fun Network Management System or JFFNMS. This program was written in PHP and combined the status polling of Nagios with the RRD graphs of cricket and MRTG. As it was written in PHP this had all the bonuses and problems of other PHP programs. It was able to reasonably easily run on Windows and Linux systems, amongst others and handled the database and SNMP parts through modules.

Maintaining a PHP program is not easy and tracking down bugs gets very difficult. There needed to be a better way and one solution was to keep PHP but use a framework such as CakePHP. While this solved some of the framework problems, it still left PHP with all its quirkyness.

Another series of searches and it was decided to start a completely new project. Given it was a rewrite, then there was no need to stay with the same langauge. Also the web framework needed to be something reasonably substancial that took care of things such as database handling, authentication and web request routing. After some research and false starts, in October 2011, RoseNMS was born based upon TurboGears.

CHAPTER 3

Installation

There are many ways you can setup RoseNMS. No single way is “correct” but there are often pitfalls depending on your particular setup. This section describes one way of install RoseNMS.

RoseNMS is basically a WSGI interface enabled program. If you understand how these sort of programs work, you are free to install RoseNMS any way you like, using your standard setup.

For the rest of us, I’ll assume you have:

- the RoseNMS egg, which contains the program;
- a working apache server with modwsgi installed;
- virtualenv which makes virtual environments and dependencies.

There is also three separate directories involved in the installation. There is absolutely no solid rules where these directories have to go, the important thing is not to mix them up.

- Baseline - This is where the python interpreter and the system files are kept. We will use /usr/local/pythonenv/BASELINE
- RoseNMS environment - Additional packages that RoseNMS needs to run will be installed here. This is the location of the specific virtualenv we will use. For the document lets call it /usr/local/pythonenv/RoseNMS_NMS
- RoseNMS home - Location of the RoseNMS files, such as a sqlite database we will use /home/rosenberg

So, now to make the various directories, part of this comes from the Virtualenv support for VirtualEnvironments page.

3.1 Baseline

```
$ cd /usr/local
$ mkdir /usr/local/pythonenv
$ cd /usr/local/pythonenv
$ virtualenv --no-site-packages BASELINE
New python executable in BASELINE/bin/python
Installing distribute...........................................
......................................................................
......................................................................
.....................done.
Installing pip................done.
```

This directory is where the WSGI server within Apache will find the python files. You will need to tell it this with a configuration parameter

code-block:: WSGIPythonHome /usr/local/pythonenv/BASELINE

3.2 RoseNMS Environment

The RoseNMS Environment is made almost the same way and will be located at /usr/local/pythonenv/rnms. It is best to install TurboGears first as it pulls in the right sort of dependencies, then install RoseNMS.

```
$ cd /usr/local/pythonenv
$ virtualenv --no-site-packages rnms
New python executable in rnms/bin/python
Installing distribute...................................................
........................................................................
........................................................................
.....................done.
Installing pip................done.
$ source rnms/bin/activate
(rnms)$ easy_install -i http://tg.gy/current Turbogears2
(lots of lines of install as things happen!)

(rnms)$ easy_install /tmp/RoseNMS_NMS-0.0.0dev-py2.7.egg
```

There will be an awful lot of work going on when you try to install RoseNMS as easy_install will go off and download all the dependent packages that are required for RoseNMS to run correctly.

3.3 Apache Configuration

The apache configuration shown below basically tells Apache where to find the baseline files and where the wsgi file is located. We have also made 3 WSGI daemons with a name of wsgid. The values given don't have to be the same but are the defaults seen in most documentation.

```
WSGIPythonHome /usr/local/pythonenv/BASELINE/
WSGIDaemonProcess example.com threads=10 processes=3 display-name=wsgid
WSGIProcessGroup example.com
<VirtualHost *:80>
  ServerName example.com
  WSGIScriptAlias /rnms /home/rosenberg/apache/rnms.wsgi
</VirtualHost>
```

3.4 WSGI File

```
import sys
prev_sys_path = list(sys.path)
import site
site.addsitedir('/usr/local/pythonenv/rnms/lib/python2.7/site-packages')

new_sys_path = []
for item in list(sys.path):
    if item not in prev_sys_path:
        new_sys_path.append(item)
        sys.path.remove(item)
sys.path[:0] = new_sys_path
#End of virtualenv support

# This adds your project's root path to the PYTHONPATH so that you can import
# top-level modules from your project path.  This is how TurboGears QuickStarted
# projects are laid out by default.
import os, sys
sys.path.append('/usr/local/pythonenv/rnms')

# Set the environment variable PYTHON_EGG_CACHE to an appropriate directory
# where the Apache user has write permission and into which it can unpack egg files.
os.environ['PYTHON_EGG_CACHE'] = '/home/rnms/python-eggs'

# Initialize logging module from your TurboGears config file
from paste.script.util.logging_config import fileConfig
fileConfig('/home/rnms/production.ini')

# Finally, load your application's production.ini file.
from paste.deploy import loadapp
application = loadapp('config:/home/rnms/production.ini')
```

CHAPTER 4

Hosts and Attributes

4.1 Zones

Zones are groupings of Hosts. They can be used for displaying a group together or for making a set of hosts visible.

4.2 Hosts

Hosts are the devices that you want to manage. They are essentially something that has an IP address (either IPv4 or IPv6) and generally would also have some sort of SNMP Agent. The Agent is not essential but is very useful as most *Attribute Types* will need SNMP. The main exeption being *Reachability*, *TCP Ports* and *NTP*.

As expected, Hosts have a management address, a name, optionally three **'SNMP Communities'_** (read only, read/write and trap) plus some other parameters such as **'Autodiscovery Policies'_**. Hosts also belong to a single *Zone*.

Hosts do not have an User but may have a default User for Attributes found during Autodiscovery. This makes sense when a single Host may service many User's services. For example, a common switch may have user A on port 1 and user B on port 2, or a particular server may have several websites owned by different users.

4.3 Attributes

An Attribute is one of the major models that is used in RoseNMS. It is effectively something that you want to monitor or track within a Host. Attributes will have RRD values to update or a status to track or perhaps both these options.

The simplest idea of an Attribute is a physical interface. This Attribute Type has counters that turn into graphs such as an error or packet rate and the operational and administrative status that change the state of the Attribute. All other Attributes are variations of this idea, but follow the same basic concept.

Besides the Host it is bound to, an Attribute can have a SLA. The particular SLA that can be assigned to an Attribute is based upon the Attribute Type. The SLA uses the last 30 minutes of data to determine if the data are within some specification. Attributes can have a poll priority. While it is not essential to set an Attribute for a host with a priority, it greatly helps with the efficiency of the poller.

Attributes with the poll priority are selected before normal Attributes. If Attributes within a host with poll priority set are down, then the remaining Attributes within that host are no polled. This means that with careful selection of prioritized Attributes, if an entire host is down then the poller doesn't waste effort attempting to get to the host. The most common Attribute Type to assign for priority is Reachability. The idea being that if you cannot ping the host, then you cannot reach it and it doesn't make sense to attempt to get any more data out of the device.

With a prioritized Attribute down for a Host, only the prioritized Attributes are polled.

CHAPTER 5

Attribute Types

The types of Attributes that can be discovered and polled is setup in the configuration. RNMS comes out with a variety of Attribute Types and you can add your own.

5.1 Apache

Polls the server status for an Apache webserver. Rnms displays the statistics that this feature exposes, such as number of workers or accesses.

5.2 APC

Polls the APC UPS devices that have SNMP enabled.

5.3 Alteon Load Balancers

Alteon Load Balancers are reasonably old devices but still in use in some places. As well as the System Information, which tracks the usual things like CPU loads and memory usage, the Real and Virtual Servers and Services are tracked for their utilisation and response times. The state of these elements is also tracked and can send alarms.

5.4 Applications

5.5 BGP Neighbors

These Attribute Types are the BGP (Border Gateway Protocol) peers. The number of advertised and received routes, as well as messages in and out are tracked. The state of the Attribute follows the state of the peer. Information comes via SNMP using the BGP peer MIB based on **RFC 1269**.

5.6 Brocade FC Ports

5.7 Brocade Sensors

5.8 Cisco 802.11X Device

5.9 Cisco NAT

5.10 Cisco PIX

5.11 Cisco Power Supply

5.12 Cisco SA Agent

5.13 Cisco System Info

An Attribute Type for the CPU or system of Cisco devices. Displays the memory and CPU statistics of the device as well as the TCP connection count.

Devices that have SNMP enabled and a Cisco enterprise for sysObjectId are polled for this system information.

Cisco Temperature Cisco Voltage

5.14 Compaq CPQ MIB

Monitors the Compaq CPQ environmental elements such as Fans, Power Supplies, Temperature and Physical Drives.

5.15 Dell Chassis

5.16 Fibre Channel Ports

5.17 IIS Webserver Information

5.18 Linux/Unix System Info

5.19 NTP

The poller checks the status of NTP servers on the host by using NTP standard control messages. If the host has at least one synchronised peer it is considered synchronised and up.

5.20 OS/400 System Info

5.21 Physical Interfaces

This is the standard ifTable interfaces that are discovered via SNMP. Almost any device that supports SNMP will have some interfaces from ifTable. The ip address table is also polled at the same time and if there is a match the addresses are applied to the interface.

The usual statistics such as octect and packet counts as well as errors are polled and the attribute follows the ifOper column of the ifTable for status.

5.22 Reachablity

All devices that have an IP address are assumed to be reachable. The Reachable Attribute Type requires fping and/or fping6 to be installed. Round trip time (RTT) and packetloss are graphed for these Attribute Types.

5.23 Sensors

5.24 Solaris System Info

5.25 Storage

Storage Attribute Types are discovered via SNMP by scanning the hrStorageTable. This table considers storage to be mainly disk drives as well as memory as “storage”. The graphs for these types show the total and used amount for the device. There is no state tracked.

5.26 TCP Ports

Systems can be port scanned to find open TCP ports. These ports then have their response time captured by connecting to the ports and, if SNMP is available, the number of connections on that port. There is also an option to check the content from the port for specific text.

For TCP ports to be autodiscovered, the nmap or nmap6 binary needs to be installed.

5.27 UPS and Lines

Queries for either a standard **RFC 1628** or a Mitsubishi UPS using SNMP. Both the device status (such as load or on-battery) and the input and output lines can be tracked.

Windows System Info

CHAPTER 6

Users and Permissions

RoseNMS uses the repoze.what method of authorization which is based upon three sets of models.

6.1 Users

The first model is the User. This is usually a person although it can be a role. A User has a username and a password and the combination of these permits access to RoseNMS. All Attributes within the system are owned by a User, which can provide them with a limited read-only access to the state of the Attribute.

A User model also has an email_address which is used to send Triggers if they are setup for it.

6.2 Permissions

For each method within each controller the second model called a Permission is used to determine access. The following permissions are defined for RoseNMS:

Name	Description
UserRO	Read-Only Access to User, Group and Permissions
UserRW	Read/Write Access to User, Group and Permissions
HostRO	Read-Only Access to Host and Attribute
HostRW	Read/Write Access to Host and Attribute
AdminRO	Read-Only Access to remaining models
AdminRW	Read/Write Access to remaining models

There is likely to me more Permissions created in future versions of RoseNMS depending on user feedback.

6.3 Groups

Groups are the glue between Permissions and Users. Users cannot have permissions granted to them directly, but belong to Groups which do have Permissions assigned to them. A User can belong to none, one or many Groups and a Group can be assigned multiple Permissions. As the relationship between a Group and a Permission is many-to-many, different Groups can have the same Permision assigned to them.

There are several pre-defined Groups within a standard installation of RoseNMS.

Group Name	Permissions
User View	UserRO
User Admin	UserRW
Host View	HostRO
Host Admin	HostRW
System View	UserRO, HostRO, AdminRO
System Admin	UserRW, HostRW, AdminRW

CHAPTER 7

Events

Events within Rnms are something that has happened. They are created by either a poller backend or a consolidator and have to be linked to a Host or an Attribute. Plain Events only have an created time, they do not have a concept of duration like alarmed Events.

CHAPTER 8

Alarmed Events

If an Event is of an EventType that permits alarms and is associated with an Attribute then the Event will be marked as an Alarmed Event. These items have a stop_time which can either be set at creation or when another Event of the same EventType and Attribute but different state (usually Up) comes along.

Event Type

All Events have an EventType. This is specific domain or aspect of an Attribute or Host. For example an interface Attribute may have an operational status change or Error count exceeded EventType. Using Event Types means it is easier to determine when the Event has been cleared, or its a duplicate Event.

Event State

All Events have a State. While an EventType will tell you the type of event, such an interface status change, or a TCP port result, the state will tell you more about the Event, such as the interface status is now Down or the TCP port is now Open.

A departure from JFFNMS is that the Event State has a StateColor which sets the color in the event viewer and the Attribute and Host maps. Previously an Events colour was determined by its Severity which was based upon the EventType. Now the state sets the colour.

CHAPTER 11

How Attribute state is determined

An Attributes state is inherited purely from the collection of active alarmed Events for that Attribute. When Rnms detects that the collection of Events has changed for an Attribute, all of the active (that is, not have a corresponding stop event or timed out) alarmed Events for that Attribute are collected and the one with the lowest priority setting has its state copied to the Attribute.

As the EventState has a link to the StateColor table, this is what also determines the colour of an Attribute in the maps.

The state of a host is similiarly determined using the same method with the exception that the active alarmed events for all Attributes within are host are evaluated.

CHAPTER 12

SNMP Traps

SNMP traps are messages that are sent from SNMP Agents, such as routers or servers to a SNMP Manager, such as RoseNMS.

For the purposes of how they are handled, a trap has the following fields:

- Source IP address
- Trap OID
- One or more VarBinds which are OID value keypairs, similair to a python dictionary.

SNMP v1 traps use a different format but they are converted to use the same format.

12.1 SNMP Trap Daemon

When the main rnmsd is started up, a thread is opened to create a receiver for SNMP traps. Traps are sent via UDP and may be high frequency or even spoofed (where the source IP address is forged). The daemon does two checks on the incoming trap.

First, the trap source IP address is checked against the known list of configured hosts. To minimize impact on the database, the result is cached in a local dictionary. If there is no Host with that IP address, the trap is discarded.

Secondly the trap is checked against duplicates. Essentially if the trap from the same source IP address with the same trap OID is seen within 5 seconds of another trap with the same properties, it is discarded. A lot of implementations send several traps for the same event and this ensures there is only one forwarded. The disadvantage is that if there is a down trap and then very soon after an up trap and they use the same OID the second one won't be processed.

12.2 How RoseNMS treats traps

RoseNMS assumes that a particular trap (identified by a trap OID) is for a specific Attribute Type. For example the ifDown traps describe something happening to a physical interface, while a temperature alarm trap is connected to either a sensor or perhaps the element (e.g. CPU). If there are multiple Attribute Types that could be applied to the same trap OID, multiple Trap Matches can be given for that OID and then the Trap Match commands (e.g. matching on description) can be used to determine which is the correct one for this trap.

At set intervals, currently 30 seconds, a trap consolidator is run on the raw traps. The input data is Host ID, trap OID and the VarBinds. The consolidator first looks for all Trap Matches that first match exactly the trap OID and then optionally secondly on the VarBinds. The point of the Trap Match is to find an Attribute and a value or multiplie values that are passed along to the backend.

Just like Pollers have several commands, Trap Matches do too that process the VarBinds of the trap to find the Attribute. Only Attributes that have the correct type (defined by the Trap Match) will be looked at.

12.3 Trap Match Commands

The following commands are used by Trap Match within the consolidator to locate the attribute.

match_index_state : The Attribute is found by examining the VarBind with the specified OID and matching against the VarBind value and the Attribute's index field.

The value is either fixed or if it is an OID the VarBinds are searched for this OID. The result can be mapped to another value before returning, such as 1=down,2=up.

CHAPTER 13

rnmsd

13.1 SYNOPSYS

rnmsd [**-hqv**] [**-c** *config*] [**-p** *pidfile*]

13.2 DESCRIPTION

rnmsd is the main RoseNMS daemon. This main program spawns off several threads to take care of the back-end of the NMS systems. For most installations, starting this program is all that is required. The threads are

- Poller - collects statistics and status on attributes
- Consolidator - converts raw events such as syslog lines into real events and alarms
- SNMP Trapd - collects SNMP traps from remote devices and stores them for subsequent Consolidator processing

13.3 OPTIONS

-c file, --config file	Read configuration settings from *file*
-d, --debug	Turn on debugging
-h, --help	Show help message and exit
-p file, --pidfile file	Write programs PID to *pidfile*
-q, --quiet	Log critical messages only
-v, --verbose	Increase verbosity of logging

13.4 SEE ALSO

- RoseNMS Documentation
- **rnms_poller** (1)
- **rnms_info** (1)

CHAPTER 14

rnms_info

14.1 SYNOPSYS

rnms_info [**-dhqv**] [**-c** *config*] [**-p** *pidfile*] *qtype id...*

14.2 DESCRIPTION

rnms_info is a tool to query the database on various models that RoseNMS contains. These queries are meant to assist administrators in troubleshooting, for example working out what Host a particular Attribute belongs to.

The info tool can query the following models: attributes, attribute types, hosts, poller sets, autodiscovery policies, slas and triggers. The second parameter is the ID of the item you want to query.

14.3 OPTIONS

-c file, --config file	Read configuration settings from *file*
-d, --debug	Turn on debugging
-h, --help	Show help message and exit
-p file, --pidfile file	Write process PID to *file*
-q, --quiet	Log critical messages only
-v, --verbose	Increase verbosity of logging

ID ID of the items you want information about

QTYPE Type of query (model) to perform, see *DESCRIPTION* for list

14.4 EXAMPLE

For example, to look at attribute #2, you would use the following commands:

```
$ rnms_info attribute 2

============================================================
Attribute            | 2: Async5 (index: 5)
```

```
------------------------------------------------------------
Host                    | 5: Cisco1700
State (admin/oper)      | down/down
Attribute Type          | 4: Physical Interfaces
Poller Set              | 42: SNMP Interface (enabled:True)
Poll Priority           | False
SLA                     | 1: No SLA
Created                 | 2012-03-11 10:34:58
Next SLA                | 2013-09-30 12:26:29.878441
Next Poll               | 2013-10-07 14:58:20.189101
------------------------------------------------------------
Fields
IP Address              | 192.168.101.1
IP Mask                 | 255.255.255.252
Peer Address            | 192.168.101.2
Speed                   | 38000
```

Each type of query will display detailed information about the requested object. There can also be cross references, so to see information about the above attributes host, you would query host 5.

14.5 SEE ALSO

- RoseNMS Documentation
- **rnms_poller** (1)
- **rnmsd** (1)

CHAPTER 15

Indices and tables

- genindex
- search

Index