NMIS9 - How to Configure High Volume SNMPTRAPS

Introduction

Configure SNMPTRAPS to scale up to 300 traps per second. The purpose of this article is to show how to configure the SNMPTRAPD to pull data from the devices to rsyslog. Then, the rsyslog collects the logs into a file to be processed by opEvents. Eventually, the opEvents will apply the filters, parsers, and actions to better use the system management, analysis, and troubleshooting based on the events.

Testing SNMPTRAPS:

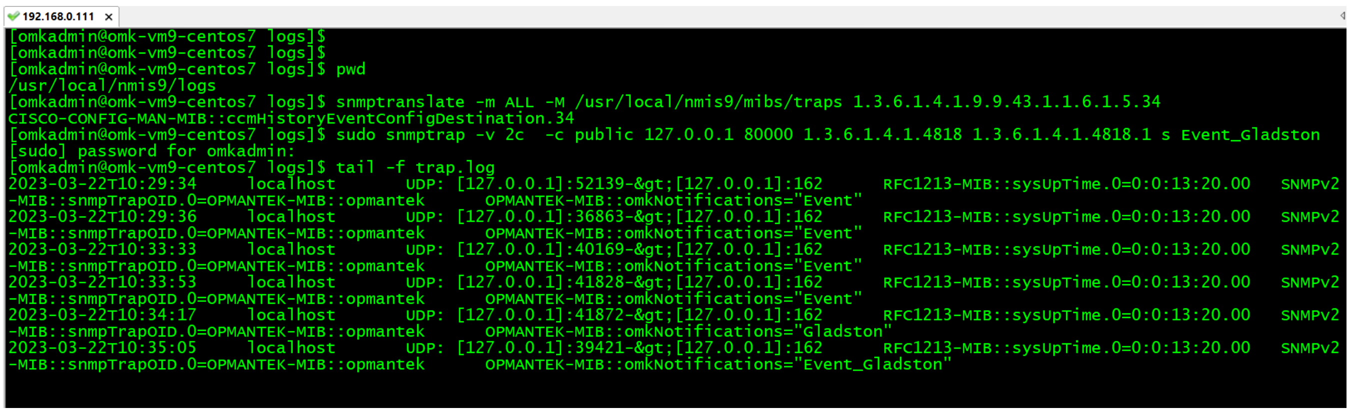
snmptranslate -m ALL -M /usr/local/nmis9/mibs/traps 1.3.6.1.4.1.9.9.43.1.1.6.1.5.34

****

RESULT: CISCO-CONFIG-MAN-MIB::ccmHistoryEventConfigDestination.34

****

sudo snmptrap -v 2c -c public 127.0.0.1 80000 1.3.6.1.4.1.4818 1.3.6.1.4.1.4818.1 s Event



Step-by-Step How to Configure the SNMPTRAPS to Forward Traps to Sylog

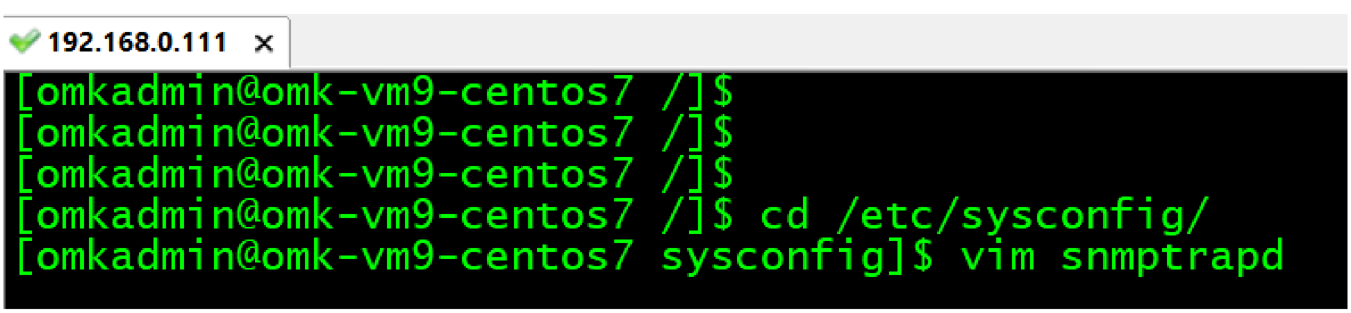
OBSERVATION: The "Ls" option will configure snmptrapd to send logs to syslog. So, "Ls2" specifically configures snmptrapd to send logs from the local2 facility. The facility is a value that indicates which process on the device generated the message.

STEP 1

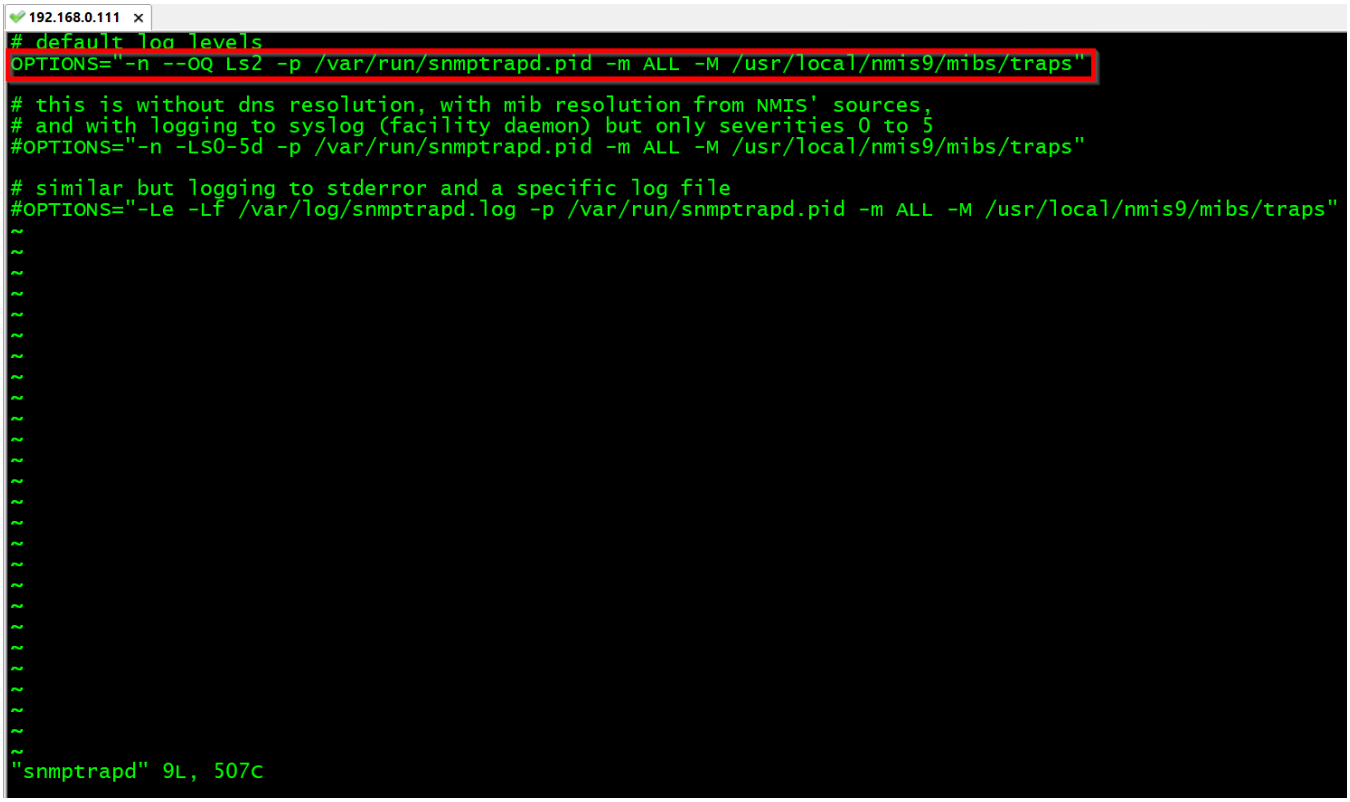
RHEL/CENTOS Linux

cd /etc/sysconfig/

vim snmptrapd



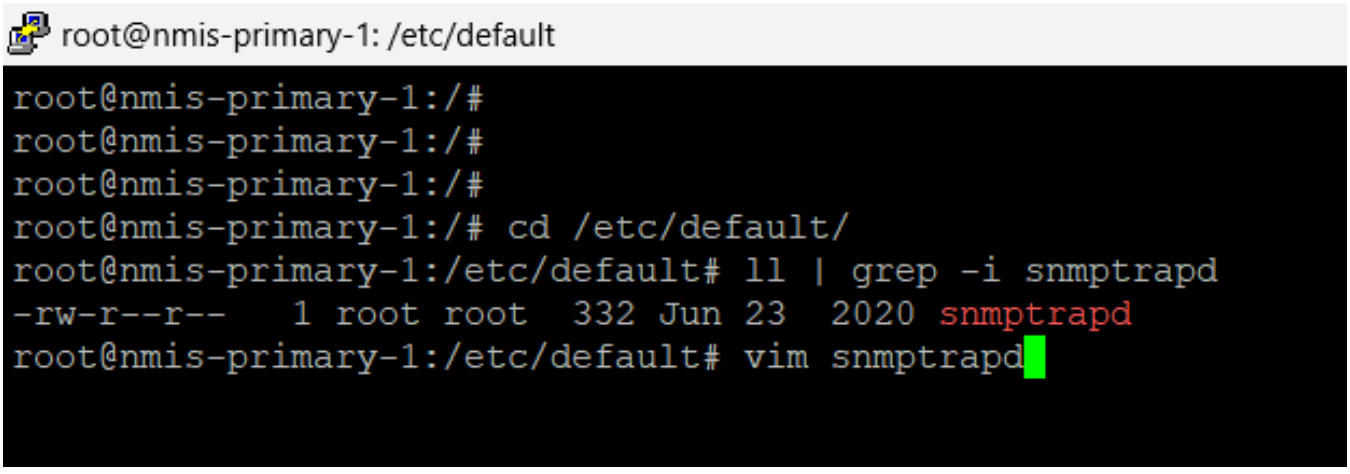
OPTIONS="-n --OQ Ls2 -p /var/run/snmptrapd.pid -m ALL -M /usr/local/nmis9/mibs/traps"



Debian/Ubuntu Linux

cd /etc/default/

vim snmptrapd



TRAPDOPTS='-n -OQ -Ls2 -p /var/run/snmptrapd.pid -m ALL -M /usr/local/nmis9/mibs/traps'
[Service]
ExecStart=
ExecStart=/usr/sbin/snmptrapd -n -OQ -Ls2 -p /var/run/snmptrapd.pid -m ALL -M /usr/local/nmis9/mibs/traps

root@opmantek-la-poller: /etc/default

```
# This file controls the behaviour of /etc/init.d/snmptrapd
# but not of the corresponding systemd service file.
# If needed, create an override file in
# /etc/systemd/system/snmptrapd.service.d/local.conf
# see man 5 systemd.unit and man 5 systemd.service

# snmptrapd options (use syslog).
TRAPDOPTS='-n -OQ -Ls2 -p /var/run/snmptrapd.pid -m ALL -M /usr/local/nmis9/mibs/traps'
[service]
ExecStart=
ExecStart=/usr/sbin/snmptrapd -n -OQ -Ls2 -p /var/run/snmptrapd.pid -m ALL -M /usr/local/nmis9/mibs/traps
```

STEP 2

We need to configure the traps to go to a specific log file for opEvents to process them. In this case, all messages that come from facility local2 will be collected into /usr/local/nmis9/logs/snmptrap.log file.

cd /etc/rsyslog.d

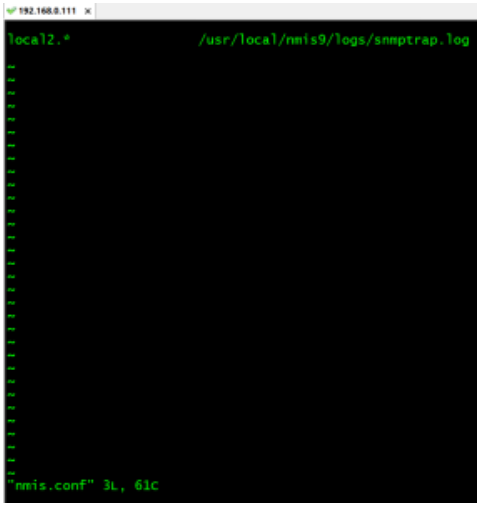
touch nmis.conf

192.168.0.111

```
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]# cd /etc/rsyslog.d
[root@omk-vm9-centos7 rsyslog.d]# touch nmis.conf
[root@omk-vm9-centos7 rsyslog.d]# ll
total 4
-rw-r--r--. 1 root root 49 Jan 13  2022 listen.conf
-rw-r--r--. 1 root root  0 Mar 22 14:13 nmis.conf
[root@omk-vm9-centos7 rsyslog.d]#
```

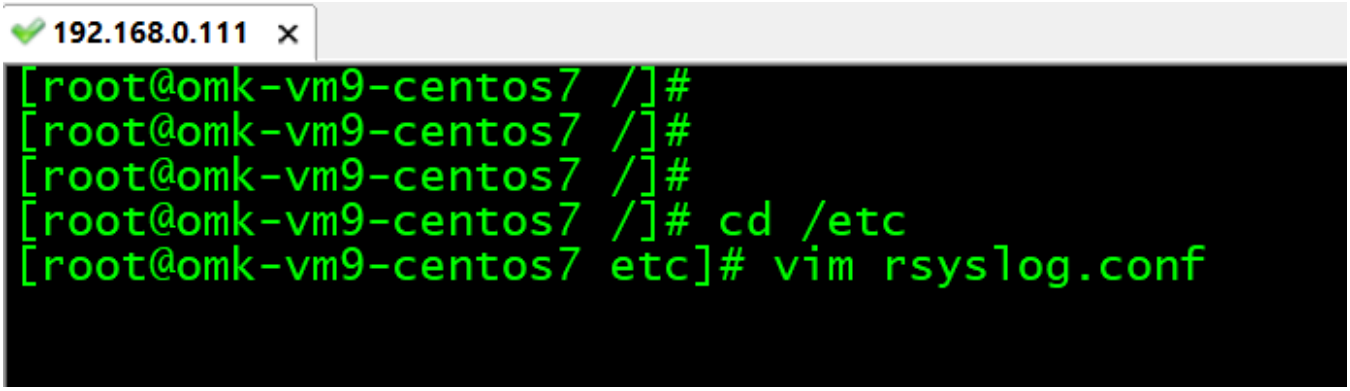
vim nmis.conf

local2.* /usr/local/nmis9/logs/snmptrap.log

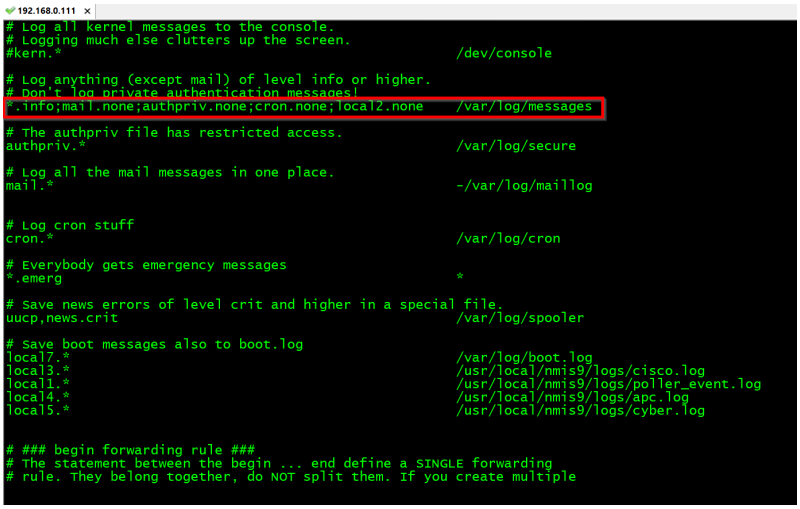


cd /etc

vim rsyslog.conf



***.info;mail.none;authpriv.none;cron.none;local2.none /var/log/messages**



STEP 3

Now, we need to inform OpEvents where the snmptrap comes from. So, we do so, informing the path where the snmptrap.log is on the /usr/local/omk/conf /opCommon.json.

****

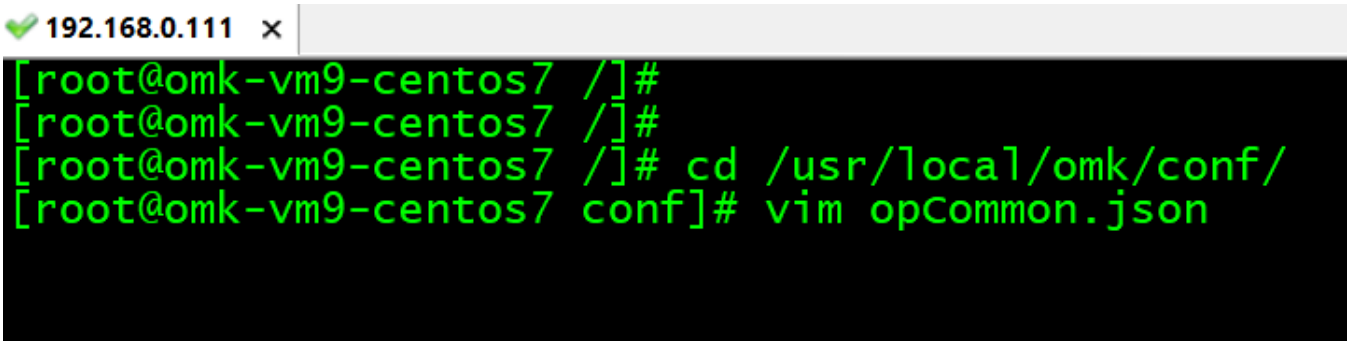
Be careful with this opCommon.json file. It is a Perl hash, so any syntax error will render the OMK Server to stop working properly.

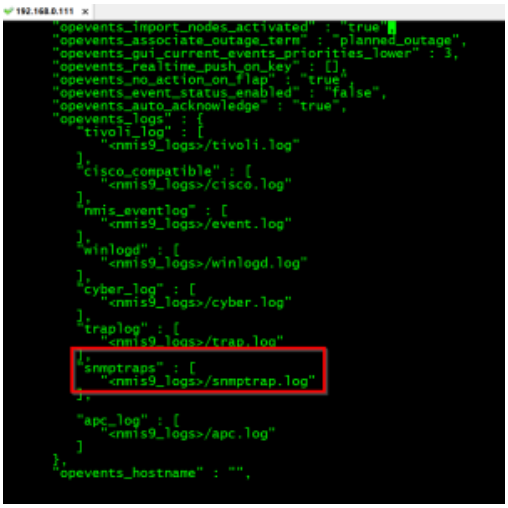
We recommend always creating a backup of this file before any changes.

****

cd /usr/local/omk/conf/

vim opCommon.json



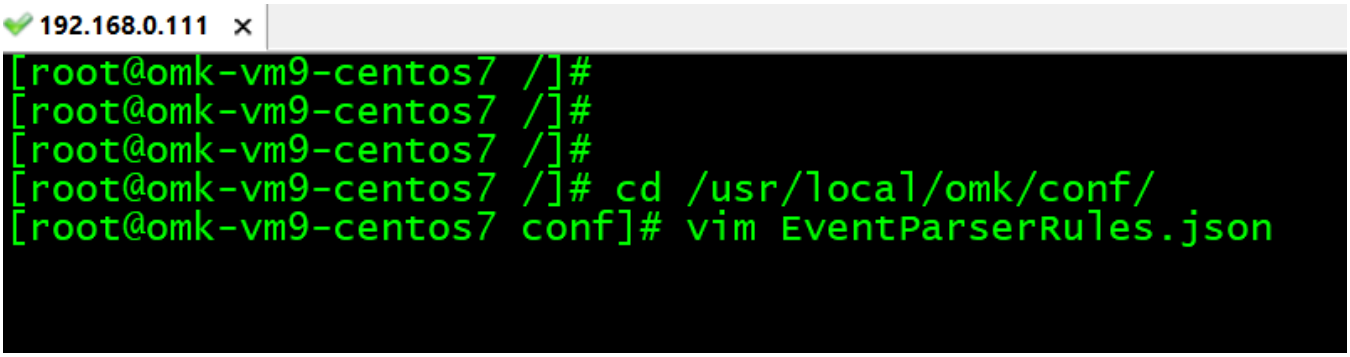


STEP 4

The parser is made on the EventPaserRules.json file. In this case, we are sending to an opEvents plugin to do the syntax translation.

cd /usr/local/omk/conf/

vim EventParserRules.json



192.168.0.111

```
{
"snmptraps" : {
        "1" : {
          "IF" : 1,
         "THEN" :["plugin(snmpTrap)"]
          },
    },
 },
  "winlogd" : {
     "1" : {
       "IF" : "\\w+\\[\\d+\\]",
       "THEN" : {
          "30" : {
             "IF" : "The\\s+(.+)\\s+service",
             "DESCRIPTION" : "The Microsoft .NET Framework NGEN v4.0.30319_X86 service entered the running state.",
             "THEN" : [
                "capture(element)"
             ]
          },
          "31" : {
             "IF" : "service entered the running state",
             "THEN" : [
                "set.state(up)",
                "set.stateful(Service)",
                "set.priority(0)"
             ]
          },
          "21" : {
             "THEN" : [
                "capture(category,user,details)"
             ],
             "IF" : "Category:([\\w\\ ]{0,256}); User:([\\w\\ ]{0,256}); (.+)$"
          },
          "32" : {
             "THEN" : [
"EventParserRules.json" 677L, 23897C                                                                                    34,1
```

STEP 5

In step 4, you added the snmptraps plugin parser rules in the EventParserRules.json file. Now, we need to copy the snmpTrap.pm file on the /usr/local/omk /conf/parser_plugins/ directory. This file is the OpEvents parser plugin. The plugin is not always needed. The traps can be collected using the event handler nmis traplog. However, the plugin can parser more complex SNMP traps.

cd /usr/local/omk/conf/parser_plugins/

vim snmpTrap.pm

192.168.0.111

```
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]# cd /usr/local/omk/conf/parser_plugins/
[root@omk-vm9-centos7 parser_plugins]# ll
total 8
-rw-rw-r--. 1 root root 2316 Apr 29  2021 README
-rw-rw-r--. 1 root root  755 Apr 29  2021 TestPlugin.pm
[root@omk-vm9-centos7 parser_plugins]# vim snmpTrap.pm
```

****

snmpTrap.pm file Download here: snmpTrap.pm

****

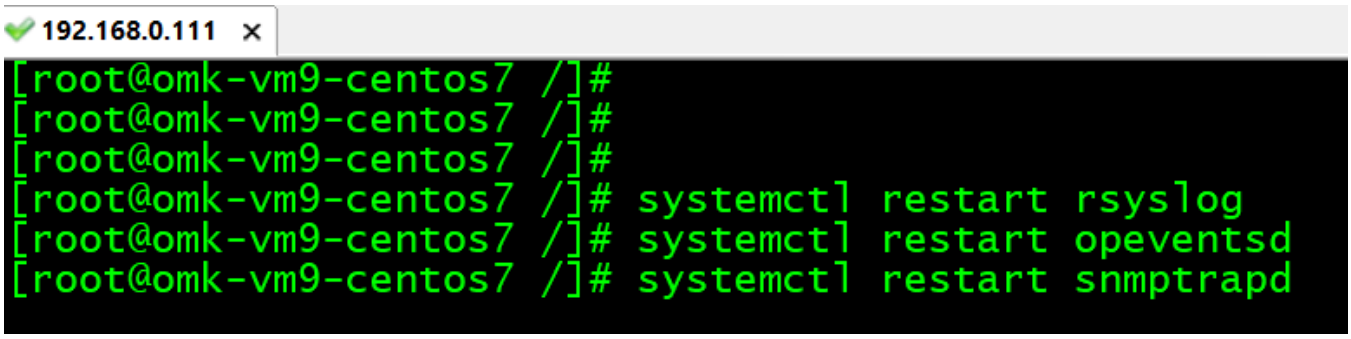
STEP 6

Restart the daemons associated.

systemctl restart rsyslog

systemctl restart opeventsd

systemctl restart snmptrapd


192.168.0.111
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]# systemctl restart rsyslog
[root@omk-vm9-centos7 /]# systemctl restart opeventsd
[root@omk-vm9-centos7 /]# systemctl restart snmptrapd