Configuring opEvents to send SNMP Traps

This page will describe the steps to configure opEvents to send SNMP traps as a proof of concept, using the SNMPTRAP commandline tool.

Pre-requisites

A copy of the OPMANTEK-MIB.mib has been obtained, this is currently in DRAFT state and is in the NMIS8 GIT source in /usr/local/nmis8/mibs/traps.

opEvents has been installed and licensed.

opEvents is already processing events and "working".

NET-SNMP is installed on the target server and the snmptrap command should be /usr/bin/snmptrap

Configuring opEvents to Send SNMP Traps

Configure the Script Action in EventActions.nmis

Before Configuring the Script Action

Where opEvents is installed, edit the file /usr/local/omk/conf/EventActions.nmis, locate the section called scripts. The default configuration file from /usr/local/omk/install includes this section:

```
        'script' => {
                'traceroute_node' => {
                        arguments => '--max-hops=20 node.host',
                        exec => 'traceroute',                        # traceroute commonly isn't in /bin
                        output => 'save'
                },
                'ping_node' => {
                        arguments => '-c 5 node.host',
                        exec => '/bin/ping',                        # but ping usually is
                        output => 'save'
                },
                'ping_neighbor' => {
                        arguments => '-c 5 event.element',
                        exec => '/bin/ping',
                        output => 'save'
                }
        },
```

Adding the Script Action

We are going to be inserting the content below into that section.

In the output below, IP_ADDRESS_OF_TEMIP needs to be replaced with the IP Address of the actual server.

```
                'send_snmptrap_poc' => {
                        arguments => '-v 2c -Ci -c OPMANTEK IP_ADDRESS_OF_TEMIP "" 1.3.6.1.4.1.4818.1.1 
1.3.6.1.4.1.4818.2.1.1 s event._id 1.3.6.1.4.1.4818.2.1.2 s event.time 1.3.6.1.4.1.4818.2.1.3 s event.date 
1.3.6.1.4.1.4818.2.1.4 s event.node 1.3.6.1.4.1.4818.2.1.5 s event.host 1.3.6.1.4.1.4818.2.1.6 s event.event 
1.3.6.1.4.1.4818.2.1.7 s event.element 1.3.6.1.4.1.4818.2.1.8 s event.state 1.3.6.1.4.1.4818.2.1.9 s event.
stateful 1.3.6.1.4.1.4818.2.1.10 s event.details 1.3.6.1.4.1.4818.2.1.11 s event.type 1.3.6.1.4.1.4818.2.1.12 s 
event.priority 1.3.6.1.4.1.4818.2.1.13 s event.level',
                        exec => '/usr/bin/snmptrap',
                        output => 'save'
                },
```

After Configuring the Script Action

Once finished you will have a script section which looks like:

```
        'script' => {
                'send_snmptrap_poc' => {
                        arguments => '-v 2c -Ci -c OPMANTEK IP_ADDRESS_OF_TEMIP "" 1.3.6.1.4.1.4818.1.1 
1.3.6.1.4.1.4818.2.1.1 s event._id 1.3.6.1.4.1.4818.2.1.2 s event.time 1.3.6.1.4.1.4818.2.1.3 s event.date 
1.3.6.1.4.1.4818.2.1.4 s event.node 1.3.6.1.4.1.4818.2.1.5 s event.host 1.3.6.1.4.1.4818.2.1.6 s event.event 
1.3.6.1.4.1.4818.2.1.7 s event.element 1.3.6.1.4.1.4818.2.1.8 s event.state 1.3.6.1.4.1.4818.2.1.9 s event.
stateful 1.3.6.1.4.1.4818.2.1.10 s event.details 1.3.6.1.4.1.4818.2.1.11 s event.type 1.3.6.1.4.1.4818.2.1.12 s 
event.priority 1.3.6.1.4.1.4818.2.1.13 s event.level',
                        exec => '/usr/bin/snmptrap',
                        output => 'save'
                },
                'traceroute_node' => {
                        arguments => '--max-hops=20 node.host',
                        exec => 'traceroute',                       # traceroute commonly isn't in /bin
                        output => 'save'
                },
                'ping_node' => {
                        arguments => '-c 5 node.host',
                        exec => '/bin/ping',                        # but ping usually is
                        output => 'save'
                },
                'ping_neighbor' => {
                        arguments => '-c 5 event.element',
                        exec => '/bin/ping',
                        output => 'save'
                }
        },
```

Configure an Action to Send SNMP Traps

Before Configuring the Policy

The default policy starts with a section 1, which is going to match any event.

```
        'policy' => {
                '1' => {
                        IF => 'node.any and event.any',
                        THEN => {
                                '10' => {
                                        IF => 'node.roleType eq "core" and event.event =~ "Down"',
                                        THEN => 'priority(+3)',
                                        BREAK => 'false'
                                },
                                '20' => {
                                        IF => 'node.roleType eq "distribution" and event.event =~ "Down"',
                                        THEN => 'priority(+2)',
                                        BREAK => 'false'
                                },
                                '30' => {
                                        IF => 'node.any and event.event eq "Node Down"',
                                        THEN => 'script.traceroute_node() AND tag.isbroken(nodedown) AND tag.
verybad(42)',
                                        BREAK => 'false'
                                },
                        },
                        BREAK => 'false'
                },
```

Adding the Policy

We are going to insert a new policy which will send an SNMP Trap for every event opEvents generates except for SNMP Traps which opEvents will generate.

The if statement here is is going to be TRUE if the contents of the event name (event.event) do NOT contain OPMANTEK-MIB

```
                                '5' => {
                                        IF => 'event.event !~ "OPMANTEK-MIB"',
                                        THEN => 'script.send_snmptrap_poc()',
                                        BREAK => 'false'
                                },
```

After Configuring the Policy

The configuration will look like below.

```
        'policy' => {
                '1' => {
                        IF => 'node.any and event.any',
                        THEN => {
                                '5' => {
                                        IF => 'event.event !~ "OPMANTEK-MIB"',
                                        THEN => 'script.send_snmptrap_poc()',
                                        BREAK => 'false'
                                },
                                '10' => {
                                        IF => 'node.roleType eq "core" and event.event =~ "Down"',
                                        THEN => 'priority(+3)',
                                        BREAK => 'false'
                                },
```

Test the EventActions.nmis file

When you have finished editing the file you can check the file by running the command **perl -c EventActions.nmis**, the result should be "syntax OK"

```
[keiths@nmisdev64 conf]$ perl -c EventActions.nmis
EventActions.nmis syntax OK
```

Testing opEvents Sending the TRAP

Installing the OPMANTEK-MIB

Copy the file OPMANTEK-MIB.mib to the target system which will be receiving the SNMP TRAPS. If this system is another vendors system, they will need to process the file into their system and confirm that it is done. To load it into an Opmantek VM copy the file to /usr/local/nmis8/mibs/traps and restart the SNMP Trap daemon "service snmptrapd restart".

Generate an Event in NMIS

The easiest way to generate an event in NMIS is to change a managed nodes host to an IP address which is unreachable and then restart the fpingd.pl.

Edit Nodes.nmis

```
    'host' => '1.2.3.4',
```

Restart the fpingd.pl

```
/usr/local/nmis8/bin/fpingd.pl restart=true
```

Monitor opEvents GUI or Logs

Check the GUI or watch the logs

```
tail -f /usr/local/omk/log/common.log
```

Change the IP address back when your done and restart fpingd.pl