

Chapter 7

Message Handler for Traps & Logs



7.1 Overview

The Message Handler is a distributed component of NetVigil which accepts syslog, SNMP traps, Windows events or any other text messages and then searches for specified patterns in these messages. When a pattern match is found, the message string is transformed and a severity assigned to it, and then forwarded to the DGE.

The Message Handler is extensible, and new data sources can be added easily into this framework. By default, the MsgHandler has built in functionality for parsing files, reading from TCP sockets, SNMP traps or Windows events (using the WMI Event Listener module - nvwmiel).

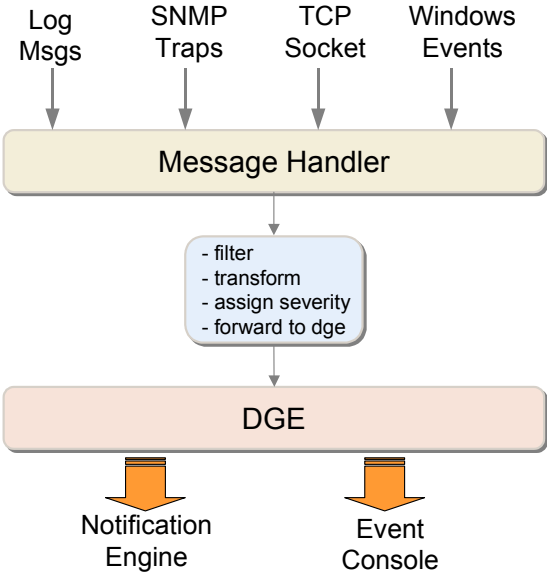


Figure 7.1 Various data sources for the Message Handler

The processed messages from the Message Handler are displayed on the NetVigil Event Manager Console and can trigger actions & notifications setup on the DGE.

Setting up NetVigil to handle traps, logs and other events requires:

- Configuring the message handler which filters, transforms and forwards the messages to the DGE, and
- Setting up filters on the DGE to display the messages on the Event Manager Console and trigger actions or notifications (described in Chapter 19, “Event Manager” on page 251)

NOTE: You must set the DGE filters using the web application under Manage -> Messages to accept messages forwarded by the Message Handler.

7.2 Starting the Message Handler

The Message Handler is installed with each DGE, and can be started using:

```
cd NETVIGIL_HOME  
etc/msgsvr.init start  
etc/msgsvr.init stop # to stop the message handler
```

It is started automatically when the DGE is started. If you want to disable this behavior, edit `etc/netvigil.init` and set `MESSAGE="N"`

7.3 Configuring the Message Handler

The MsgHandler has its own configuration file which specifies the different data sources. This config file `00_config.xml` is stored in the `NETVIGIL/etc/messages/` directory by default. You must restart the message handler after making any changes to this configuration file.

etc/messages/00_config.xml

.....

```
<message-handler>  
  
  <classes>  
    <method type="file"  
      class="com.fidelia.emerald.message.sources.FileReader"/>  
    <method type="socket"  
      class="com.fidelia.emerald.message.sources.SocketReader"/>  
    <method type="snmp-trap"
```

```

        class="com.fidelia.emerald.message.sources.SnmpTrapSource"/>
    <method type="winevt"
        class="com.fidelia.emerald.message.sources.WindowsEventSource"/>
</classes>

<ruleset-defaults type="*">
    <action>ignore</action>
</ruleset-defaults>
<ruleset-defaults type="socket">
    <action>accept</action>
    <severity>ok</severity>
    <show-message>true</show-message>
    <auto-clear>300</auto-clear>
    <transform>${device_name} ${raw_message}</transform>
</ruleset-defaults>

<source type="file" name="syslog">
    <enabled>>false</enabled>
    <input>/var/log/messages</input>
</source>
<source type="file" name="apacheErrLog">
    <enabled>>true</enabled>
    <input>/apache/logs/httpd.error</input>
</source>

<source type="socket" name="ism">
    <enabled>true</enabled>
    <port>7666</port>          <!-- port for incoming connections -->
    <connections>4</connections> <!-- maximum concurrent connections -->
    <timeout>120</timeout>    <!-- idle timeout, in seconds -->
    <user-name>ismuser</user-name><!-- username to use for login -->
    <password>fixme</password> <!-- password to use for login -->
</source>

<source type="trap" name="trap162">
    <enabled>true</enabled>
    <port>162</port>
    <performHostnameLookup>>false</performHostnameLookup>
</source>

<source type="winevt" name="windowsEvents">
    <enabled>true</enabled>
    <address>192.168.1.160</address>
    <port>7668</port>
    <username>wmiel</username>
    <password>fixme</password>
    <timeout>60</timeout>    <!-- socket timeout, typically 60sec -->
    <severity>warn</severity> <!-- * or 1,2,3,4,5 or info|warn|error -->
    <hosts>
        empire
        whistler
        winserv myuser mypassword
    </hosts>
</source>

```

```

    </hosts>
  </source>
</message-handler>

```



The different configuration parameters in the above configuration file are described below:

Table 7.1

Parameter	Description
ruleset-defaults	A default ruleset for pattern matching. See the description of each element in the section on Adding Rules below.
source	The type is one of the listed 'classes' (file, socket, trap or winevt). See the description of each source type in the section below.

7.4 Configuring the Message Sources

There are currently 4 types of message sources that can be configured in the Message Handler. These types are:

1. File - for text files
2. Trap - for SNMP traps
3. Socket - for reading from a TCP socket
4. WinEvt - for Windows events using `nvwmie1`

The “name” parameter in the source configuration is matched against the corresponding ‘name’ parameter in the rule definitions to control which rules are applied against which message sources.

Detailed instructions on each of these sources is provided further below (see “Processing Log files” on page 93)

Adding Custom Message Sources

Users can extend the Message Handler to handle additional message sources very easily by creating additional configuration files and storing it in the plug-ins directory under `NETVIGIL/plugin/messages/`. You can create additional log files to be monitored, additional trap handlers running on different ports, or other TCP sockets to accept text



streams. For details on how to extend NetVigil using the plug-in architecture, please look at Section 27.5, “Extending the Message Handler” on page 349

7.5 Adding Rulesets

The Message Handler searches and loads all rule files in the NETVIGIL/plugins/messages and the NETVIGIL/etc/messages/ directories on startup. These rule files have the naming format *xx_rule_yyyy.xml*, and are loaded in sequential order sorted by name.

These rules are used to parse the log messages using regular expressions, and extract the different fields such as device name, test, log message and then decide if the message should be accepted or dropped because it is not interesting.

Finally, the incoming log messages are transformed into a different output format based on the rule transformation element.

Once the message is transformed, it is forwarded to the specified Data Gathering Engine (DGE), where it is displayed on the Event Manager Console, and optionally, trigger an action.

7.5.1 Rule Specifications

The default format for a rule is:

```
<NetVigil>
  <message-handler>
    <ruleset type="type_name" name="source_name">
      <rule>
        <description>descriptive_text</description>
        <pattern>regular_expression</pattern>
        <action>match_action</action>
        <mapping>
          <field name="field_name_1" match="match_index_1"/>
          <field name="field_name_2" match="match_index_2"/>
          [...]
          <field name="field_name_n" match="match_index_n"/>
        </mapping>
        <severity>severity_name</severity>
        <show-message>true</show-message>
        <auto-clear>600</auto-clear>
        <transform>new_message</transform>
      </rule>

      <rule>
        [...] <!-- multiple rules -->
      </rule>
    </ruleset>
  </message-handler>
</NetVigil>
```

```

    </rule>
  </ruleset>
</message-handler>
</NetVigil>

```



where

Element Name	Description
type	one of the types defined in 00_config.xml (file, socket, trap, winevt)
name	matches the source 'name'. It can be * in which case its rules are checked before any other rulesets.
description	free-form text describing the incoming message (optional)
pattern	perl5 (hence 'oro') compatible regular expression. The match assumes 'ignorecase' is set (case is ignored).
action	one of: accept, reject
mapping.field.name	one of: device_name, device_address or a unique word
mapping.field.match	1 .. n. this corresponds to one of the match items from regular_expression
severity	one of: ok, warning, critical, unknown
show-message	Set to true or false. The remote DGE will not display the message on the console, but can still be used to trigger an action and generate reports.
auto-clear	Optional. Automatically removes the message from the console after the specified number of seconds.
transform	Converted message which is sent to the DGE.

You can have a default rule that matches everything using:

```
<pattern>.*</pattern>
```

You can log each message that comes in before the rules are applied by enabling debug level logging for the message handler in the etc/log4j.conf file.



Sample rule for sshd

```

<NetVigil>
  <message-handler>

  <ruleset type="file" name="syslog">
    <rule>
      <description>SSH: Break-In Attempt as ROOT</description>
      <pattern>:\d+\s+(\S+)\s+(\S+)\s+[\d+]:\s+.*\s+root\s+from\s+(.*)\s+ssh2</pattern>
      <action>accept</action>
      <mapping>
        <field name="device_name" match="1"/>
        <field name="process_name" match="2"/>
        <field name="remote_host" match="3"/>
      </mapping>
      <severity>critical</severity>
      <show-message>>true</show-message>
      <auto-clear>600</auto-clear>
      <transform>${process_name}: break-in attempt as "root" from
${remote_host}</transform>
    </rule>
  </ruleset>

</message-handler>
</NetVigil>

```

❑ NOTE:

1. One of “device_name” or “device_address” field is required. If one is specified, the other can be optional. If neither is specified, or there is no match found, then the message is dropped (because there is no way to match the message with a provisioned devices).
2. Within the “<transform>” section, the variables (\${foo}) correspond to fields defined in “<map>” section. If a variable specified was not defined before, or was not matched, the message is dropped
3. Even if the value in “<transform>” is specified on multiple lines for readability, the final message is on a single line. The original message is still accessible via the \${raw_message} variable. If no value is specified for this attribute, or the attribute is missing, it defaults to the message as it was originally accepted (i.e. <transform>\${raw_message}</transform>)
4. You can specify a special name “*” for a source type, which will be applicable to all sources of that type. The rules in this set are checked before any other rules.

5. If there is a ruleset with name “_default”, it is used *after* all other rules have been checked and there was no match
6. If the ruleset does not extract the “TIME” string, then the system uses the default timestamp. However, the user can extract the “TIME” string (free format string), and the message handler will attempt to convert the free form text string into the proper time syntax

As the text messages are collected by the various data sources, they are matched against all rules (sorted by the order they are read) in all files (sorted by name). At the first match (either accept or reject), no further processing is done. The message is transformed and then forwarded to the DGE.

It is important to organize the rules so that the majority of the messages are matched early on (either accepted or rejected) for better performance.

In absence of a <ruleset-defaults> entry, the following defaults are used:

match_action	accept
severity_name	ok
new_message	\${raw_message}
show_message	true
auto_clear	false

7.5.2 Regular Expressions

The patterns specified in the rulesets are Perl-5 compatible regular expressions. The standard meta characters used in regular expressions are:

Meta character	Meaning
^	Match beginning of the line
\$	Match end of the line (newline)
[]	Character class (match any character within [])

Meta character	Meaning
.	Match any character
\d	Match any digit: [0-9]
\D	Match any non-digit: [^0-9]
\s	Match any whitespace (tab, space)
\S	Match any non-whitespace character
\w	A word character [A-Za-z_0-9]
X?	Match X zero or one time
X*	Match X zero or more times
X+	Match X one or more times
()	Grouping to extract fields

As an example, to match the string:

Login failure for superuser from 128.121.1.2

you can use the following regular expression:

```
\s+Login\s+failure\s+for\s+(\S+)\s+from([0-9.]+)$
```

The parens allows you to extract the username and the IP address as \$1 and \$2 fields respectively.

7.6 Provisioning Devices on the DGE

The user can also filter messages on the DGE based on the host name and the “source name”. If the user wants to trigger actions based on an incoming message, that also needs to be configured on the DGE.

The DGE is configured via the NetVigil web interface and going to `Manage -> Messages`. This is described in detail in Chapter 16, “Manage NetVigil” on page 199.

7.7 Processing Log files

7.7.1 The “File” message source

The Message Handler “file” source type has the ability to “watch” text files for specific patterns (only new lines that are added to the file are processed and not the existing text).

As an example, the following entry will monitor the following file
/var/log/messages:

```
<source type="file" name="syslog">  
  <enabled>true</enabled> <!-- to temporarily disable -->  
  <input>/var/log/messages</input>  
</source>
```

The 'input' parameter is set to the name of the text file. You must add a new 'FILE' entry for each text file that you would like to monitor. To avoid your changes getting overwritten during NetVigil upgrades, you should add these entries as plug-ins in nn_src_yyy.xml config files in the NETVIGIL/plugins/messages/ directory.

7.8 Processing SNMP Traps

7.8.1 SNMP Traps Overview

Various router/switch/network appliances and applications have the ability to send SNMP traps to indicate some event has transpired. NetVigil has the ability to accept such SNMP traps from devices it is monitoring and display these messages as well as trigger an action using the Message Handler framework, when a pattern is matched.

In order for NetVigil to process SNMP traps, the end devices need to be configured to send traps to the host running the NetVigil Message Handler. Please refer to the respective documentation of the router, server or application to find out how to configure trap destinations.

7.8.2 The "Trap" message source

The "trap" message source handles SNMP traps and by default it is configured to run on port 162. The configuration entry in 00_config.xml for the Message Handler is:

```
<source type="trap" name="trap162">  
  <enabled>true</enabled>  
  <port>162</port>  
  <performHostnameLookup>>false</performHostnameLookup>  
  <relay oid=".1.3.6.1.4.1.10844.1.1.255.1">  
    <destination host="localhost" port="9991"  
      communityId="public"/>  
    <destination host="127.0.0.1" port="9991"  
      communityId="public"/>  
  </relay>  
</trapHandle oid=".1.3.6.1.4.1.10844.1.1.255.1">
```

```
        <script>/tmp/trapreceived.sh</script>
    </trapHandle>
</source>
```

You can choose to run the trap handler at an alternate (UDP) port other than the standard port 162 by modifying the `port` parameter. In that case, make sure to specify the alternate destination port number on remote devices that will send SNMP traps.

To avoid your changes getting overwritten during NetVigil upgrades, you should add these entries as plug-ins in `nn_src_yyy.xml` config files in the `NETVIGIL/plugins/messages/` directory.

The `performHostnameLookup` parameter controls whether the trap handler will attempt to resolve the host name of remote hosts when a trap is received. As slow DNS resolutions may impact performance, the default option disables this feature.

Once any of these values have been changed, the Message Handler will need to be restarted before the change is applied. At this point the trap handler should be ready to accept SNMP traps.

7.8.3 Relaying SNMP Traps

In certain cases, you may wish to “relay” the SNMP traps to another application. You can relay all or selected traps to one or more hosts:

```
<source type="trap" name="trap162">
  <!-- forward fidelia traps to hostA -->
  <relay oid=".1.3.6.1.4.1.10844.*">
    <destination host="192.168.1.1" port="162"
      communityId="public"/>
  </relay>

  <!-- forward all other to hostB and hostC -->
  <relay oid="default">
    <destination host="192.168.2.2" port="162"
      communityId="public"/>
    <destination host="192.168.5.5" port="8162"
      communityId="secret"/>
  </relay>
</source>
```

In the above example, all enterprise traps for Fidelia Technology MIB with prefix `.1.3.6.1.4.1.10844` is relayed to a management agent (specified in `destination` element) running on host `192.168.1.1`, on UDP port 162. Note the use of the `*` as wildcard in the `oid` parameter. If

you wish to forward only specific traps, you can use exact OID. The second `relay` configuration block has an `oid` value `default`, which has special meaning and covers any OID not explicitly specified in other `relay` blocks. The `default oid` is optional and if not specified, in the absence of a matching `relay` block, the trap will not be forwarded to any other host. In this case all traps are forwarded to two hosts, each with different port and community string.

7.8.4 Passing SNMP Traps to External Scripts

The trap handler also allows SNMP traps to be passed to external scripts, which can further process them:

```
<source type="trap" name="trap162">
  <!-- forward fidelia nodeDeleted traps to a script -->
  <trapHandle oid=".1.3.6.1.4.1.10844.1.1.255.2.1">
    <script>/usr/bin/nodeDeleted.pl</script>
  </trapHandle>
</source>
```

The same rules for wildcard (*) and default OID as `relay` configuration applies to `trapHandle` configuration. Upon match, the specified script is executed and trap information is made available via standard input (STDIN) in the following format, one entry per line in sequential order:

```
remote_device host_name
remote_device ip_address
system.sysUpTime.0 uptime
snmpTrap.snmpTrapEnterpriseOID enterprise_oid
varbind_oid1 varbind_value1
varbind_oid2 varbind_value2
[...]
varbind_oidN varbind_valueN
```

If DNS resolution is disabled, or DNS resolution failed, `host_name` will be same as `ip_address`. `uptime` represents number of seconds since remote agent was started or initialized.

7.8.5 Loading Enterprise MIBs for SNMP Traps

NetVigil monitors use OID values to poll for results and does not need MIB files. However, if you are using the SNMP trap handler, and wish to create severity mappings using OID name, you will need to load the MIB definition into the trap handler. To load a new SNMP MIB file, simply copy the MIB file to `NETVIGIL_HOME/lib/mibs` and run:

```
NETVIGIL_HOME/utils/loadSnmpMIBs.sh
(NETVIGIL_HOME/utils/loadSnmpMIBs.cmd on Windows)
```

Then restart the DGE to load the new MIBs.

Cisco MIBs that have the “.my” extension needs to be copied into the lib/mibs directory and run through utils/loadSnmpMIBs.sh to convert them into a suitable format for the DGE. There are regular mib and trap files that can be downloaded from the Cisco site. For example, when a generic linkup/linkDown trap is accepted by NetVigil and IF-MIB is compiled/loaded into the trap handler, here is how they look on the message windows.

```
TRAP : 103464683 204.0.80.117 (linkDown) 1:ifIndex.2=2; 2:ifDescr.2=Ethernet0/0;
3:ifType.2=6;

TRAP: 103464882 204.0.80.117 (linkup) 1:ifIndex.2=2; 2:ifDescr.2=Ethernet0/0;
3:ifType.2=6;
```

Also note that utils/loadSnmpMIBs.sh will complain if there are missing dependencies. When you add one particular MIB(.my) file, look at the IMPORTS directives, copy the MIBs references into that directory, look in those files and so on. For example, in order to get the IF-MIB loaded, the following MIB definition files needed to be added:

```
IF-MIB
RFC1213-MIB
SNMPv2-SMI
SNMPv2-TC
SNMPv2-CONF
SNMPv2-MIB
```

7.9 Processing data from the socket interface

7.9.1 The “Socket” message source

The ‘socket’ message source allows any external tool to send text messages over a TCP socket. These messages are then processed using the corresponding rules.

To configure a new TCP socket message source, create the following entry in a nm_src_yyy.xml file in the plugins/messages directory:

```
<source type="socket" name="nvmsgsocket">
  <enabled>true</enabled>
  <port>7666</port> <!-- TCP port -->
```

```
<connections>4</connections>  
<timeout>120</timeout> <!-- idle timeout in secs -->  
<user-name>ismuser</user-name>  
<password>fixme</password>  
</source>
```

The various parameters control the number of concurrent connections, port number, login username and password for the socket interface.

In order to connect and send messages over the TCP socket, the client must first log into the socket source using the configured username and password. After logging in, the client can send text strings in free text format (terminated with a `\r\n`).

The commands sent by a client and responses sent back by the server must adhere to the following formatting conventions:

Client Command Format

1. Each client command is composed of a single line of text terminated by a newline character. A carriage return followed by a newline (`\r\n`) is considered to be the same as a newline character (`\n`) alone.
2. Client commands may or may not require additional parameters. Each parameter consists of values, separated by 'pipe' symbol (`|`).
Example `command_name value1 [| value2 | value3 ..]`.
3. Pipe symbol (`|`) is not permitted as part of the value.
4. For each client command, the server will respond with a response code indicating success or failure, and optionally some descriptive text indication actions taken.
5. Command names are NOT case sensitive.
6. Parameters/values for any command must appear in exact order following the command. If a value is not applicable or existent for a particular command, an empty value (`| |`) should be provided.

Server Response Format

The server will always respond (to client initiated commands/requests) with text of the following format:

```
<status code> [optional informative text]
```

where `status code` is one of:

OK which indicates the command/request was successful

ERR which is indicative of failure to execute the request

Client Commands

Login

Provide authentication information to the server. This username and password are specified in the dge.xml configuration file.

```
Login <login_id> | <password>
```

Logout | Quit

End a login session.

```
LOGOUT
```

7.9.2 Input Stream Monitor (ISM)

If the socket source “name” is set to “ISM”, then you can insert pre-processed log messages which will NOT be processed for any rules, and forwarded to the DGE directly.

After logging into the socket ISM source using LOGIN, you insert a processed text message using the following command:

```
Message.insert device_name | device_addr | test_name | test_serial  

    | date_time | severity | message
```

where

- `severity` is one of "ok", "warning", "error", "critical", signifying level of urgency for the message. The default severity is "ok".
- `message` is free flowing text, up to 255 characters.

`test_name` and `test_serial` can remain blank, and the message would be inserted as a general message for the device, identified by `device_name` and/or `device_addr`.

Each parameter is separated by a ‘|’ character.

7.10 Processing Windows Events

7.10.1 The NetVigil WMI Event Listener (nvwmiei)

The NetVigil WMI Event Listener (nvwmiei) is an agent that runs on any one windows host in your workgroup or domain, and retrieves events from all other windows hosts that are part of the domain or workgroup.

Windows events are usually classified in 3 categories - application, system and security and the severity ranges from error, warn to info.

Installing the WMI Event Listener

To install the Event Listener, download the nvwmiei-xx.exe package from support.fidelia.com and run it on a Windows XP/2000/2003 server. Note that due to Windows built in security features, you must install the software as a domain administrator, and provide the domain administrator username and password during installation so that the Event Listener can query other hosts in the domain using WMI.

If you have XP SP2 running on the target machines, you will need to either disable the Internet Connection Firewall (ICF) or allow the host running nvwmiei to access the machine. You can do this by going to the Start -> Control Panel -> Windows Firewall.

7.10.2 The “WinEvt” message source

The WinEvt message source uses the NetVigil WMI Event Listener module (see above) to get events from Windows hosts and then process them using the defined rulesets for the message handler.

```
<source type="winevt" name="windowsEvents">
  <enabled>true</enabled>
  <address>192.168.1.160</address>
  <port>7668</port>
  <username>wmiei</username>
  <password>fixme</password>
  <defaultDomainUser>.\Administrator</defaultDomainUser>
  <defaultDomainPassword>myDomainPassword</defaultDomainPassword>
  <timeout>60</timeout> <!-- socket timeout, typically 60sec -->
  <severity>warn</severity> <!-- * or info|warn|error -->
  <hosts>
    empire
    whistler
    winserv myuser mypassword
```

```
</hosts>
</source>
```

The different fields are:

Element Name	Description
type	must be set to "winevt"
name	Can be any text name to identify this source in the rulesets.
address	IP address of the host running the <code>nvwmie1</code> Event Listener software
port	TCP port number for <code>nvwmie1</code> , should be set to 7668
username / password	For logging into the <code>nvwmie1</code> agent.
defaultDomainUser	The default username and password to be used for your windows domain or workgroup. Specify the username using the syntax <code>.\User</code> or <code>domain\User</code>
timeout	Close the connection to the <code>nvwmie1</code> agent if it is unreachable for more than these many secs.
severity	this is the severity of the windows events that should be retrieved. One of <code>info warn error</code>
hosts	This is a list of hosts, one per line (and optional username and password) that should be monitored for their event logs. Note that all the Windows hosts provisioned in NetVigil are automatically added to this list every 10 minutes.

7.11 Example

This is an example of how to setup NetVigil to receive an alert when there is a trap sent by a Netscreen firewall for a UDP flood alert.

1. Add a rule in your ruleset definition file, e.g. add the following text to the `plugins/messages/00_rule_traps.xml` file:

```
<NetVigil>
<message-handler>
<!-- udp flood rule -->
<ruleset type="trap" name="162">
```

```
<rule>
<description>Netscreen: UDP Flood Attack</description>
<pattern>TRAP:\s+\S+\s+(\S+)\s+(\S+)\s+\.1\.3\.6\.1\.4\.1\.3224\
.1\.4:200\s+1:[^=]+=12;\s+2:[^=]+(=[^:]+:\s+)?(.*);</pattern>
<action>accept</action>
  <mapping>
    <field name="device_name" match="-1"/>
    <field name="device_address" match="1"/>
    <field name="alert_text" match="3"/>
  </mapping>
  <transform>${alert text}</transform>
  <severity>warning</severity>
  <show-message>true</show-message>
  <auto-clear>300</auto-clear>
</rule>
</ruleset>
</message-handler>
</NetVigil>
```

2. Provision the firewall device into NetVigil as an end user by going to manage -> devices -> create a device. There is no need to create any specific test for this purpose.
3. Make sure you are accepting SNMP traps from this device by going to Manage -> Messages and add this device to the "accept" list or else select "accept all events"
4. If the device is provisioned under a name and/or address that is not same as the source of incoming traps, you will need to add this address under manage -> messages -> device aliases
5. Finally, apply an action profile to this type of event. Click on manage -> actions -> assign to events, enable "select message types" next to the firewall device, and select the same event (as above). If you didn't want to individually select message types (i.e. only filter by type that you accept), you could use manage -> messages -> event notification, and apply an action profile for "action(s) in the selected profile should be executed". This will cause this action profile to be executed for all matched message events.

This example triggers the following email notification:

```
From: netvigil@fidelia.com
Date: Wed, 27 Apr 2005 08:03:41
To: root@fidelia.com
Subject: [NetVigil] fw00.dnvr01/Warning: Netscreen: UDP Flood
Attack
Event Match Notification from NetVigil:
```

```
Department Name      : Acme_Company
Device Name          : fw00.dnvr01
Device Address       : 204.0.80.43
Event Source         : trap/162
Current Severity     : Warning
Test Time            : April 27, 2005 8:03:41 AM MDT
Transformed Message :
Port Scan Attempt from 213.46.8.202 to 204.0.80.49 protocol 6 (No
Name) (2005-4-27 08:46:38)
```

