

Chapter 4



Configuration for Actions & Notifications



4.1 Overview

NetVigil has a very flexible action and notification engine which can send email, alphanumeric pages, SNMP traps and be extended to run any other program (such as restarting a process or deleting log files, etc.). The module has a built in escalation engine so that notifications can be sent to different people as devices remain in a non-OK state for an extended period of time. The notifications can also be customized based on the time of day and week by applying custom 'schedules' to the action profiles (see the "NetVigil Web User Guide" for details).

All notifications are handled by the DGEs themselves (and not a centralized BVE), which allows for a highly fault tolerant and distributed architecture.

Details on setting up Actions and Notifications via the web interface is described in the "NetVigil Web User Guide". This chapter deals with the system wide configurations required for setting up Actions and Notifications.

4.2 E-mail Notifications

There are two types of email notifications that can be sent:

- Regular email - for sending to a regular mail account
- Compact email - for sending to a mobile phone using email

Email Relay servers

Configuration File	Affected Components	Affected Operating Systems
NETVIGIL_HOME/etc/netvigil.xml	Monitor, Report Server	Linux, Solaris, Windows

The DGE needs to know which E-mail server(s) they should use to send notifications or reports via e-mail. Edit the following section:

```
<email-servers>
<host name="my_mail_server" priority="10"/>
</email-servers>
```

Change `my_mail_server` to the fqdn of your local E-mail server or the E-mail server that you use for sending outgoing E-mail. If you have more than one E-mail server, you may add additional servers with a different priority value. The NetVigil component responsible for sending mail will start with the E-mail server with the lowest priority, and if it is unable to reach that server, it will move onto the next server on the list until the notification has been sent out successfully. You should make sure that the E-mail server(s) is configured properly to allow NetVigil to relay E-mail to any E-mail address. (Please refer to your E-mail server's administration guide for instructions on how to accomplish this.)

Setting Time Zone for E-mail Notifications:

If you are using Windows version of NetVigil, use notepad to edit “NETVIGIL_DIR\bin\monitor.lax” and at the bottom of the file, add the following line:

```
user.timezone=America/Los_Angeles (e.g.for Pacific time zone)
```

Once the entry has been added, save the file and restart the “Data Gathering Element” using the service controller.

4.3 Alphanumeric Paging

NetVigil can send alphanumeric messages to a TAP/IXO pager using a modem attached to the DGE. Note that each DGE has one or more locally attached modems, which ensures maximum redundancy and fault tolerance in a distributed environment.

□ **To configure NetVigil for alphanumeric paging:**

1. Add modem configuration information to `NETVIGIL_HOME/etc/netvigil.xml` on the DGE that will send the paging notifications. See Section 4.3.1, “Modem Configuration” on page 49 for details.
2. On the same DGE, add paging central information for the paging service provider to `NETVIGIL_HOME/etc/netvigil.xml` as described in Section 4.3.2, “Paging Central Configuration” on page 50.
3. Create Action Profiles that use alphanumeric paging as described in the “NetVigil Web User Guide”, and assign them to tests that are run by this DGE.

4.3.1 Modem Configuration

You can have multiple modems attached to a DGE. For each modem that’s attached to the DGE, add a `modem-config` section to the DGE’s `NETVIGIL_HOME/etc/netvigil.xml` file. If multiple modems are configured, they are used in the order specified by their `device priority` parameters (the lower the number, the higher the priority). For each modem, set the following:

Parameter	Purpose
<code>sender id</code>	The phone number used to identify this modem when sending a page. You can set it to any phone number representing this DGE.
<code>device priority</code>	This modem’s priority with respect to other modems attached to the DGE. The lower the value of this parameter, the higher the modem’s priority. When sending a page, NetVigil uses the highest-priority modem that is available.
<code>port</code>	The port through which this modem communicates. For UNIX DGEs enter a port in the format <code>/dev/ttySn</code> where <code>n</code> is 0,1,2. For Windows DGEs use the format <code>COMn</code> where <code>n</code> is the number of the COM port.
<code>speed</code>	The modem’s transmission speed, expressed in bits per second.
<code>parity</code>	The type of parity checking, if any, used by this modem. Possible values are <code>even</code> , <code>odd</code> , and <code>none</code> .

Parameter	Purpose
databits	The number of data bits transmitted in each series. Possible values are 7 and 8.
stopbits	The number of bits used to indicate the end of a byte. Possible values are 1, 1.5, and 2.

Sample netvigil.xml modem configuration

```

.....
<modem-config>
  <sender id="3035557777"/>
  <device priority="10">
    <port>/dev/ttyS0</port> <!-- /dev/ttyS or COMn -->
    <speed>9600</speed> <!-- bps -->
    <parity>none</parity> <!-- none, odd, even -->
    <databits>8</databits> <!-- 8, 7 -->
    <stopbits>1</stopbits> <!-- 1, 1.5, 2 -->
  </device>
</modem-config>
.....

```

4.3.2 Paging Central Configuration

Every paging service provider has its own central number and modem pool configuration. For each paging service provider that will be used, add a `paging-central` child element to the `alpha-pager` element of the DGE's `NETVIGIL_HOME/etc/netvigil.xml` file. For each service provider, set the following:

Parameter	Purpose
name	A name that uniquely identifies this service provider to the DGE.
number	The number the DGE must dial to reach paging central, including any prefixes. You can find many paging central phone numbers at http://www.notepager.net/tap-phone-numbers.htm or similar sites.
speed	The highest speed supported by the service provider. Possible values are 0 (110bps), 2 (300bps), 4 (1200bps), 5 (2400bps), 6 (4800bps), and 7 (9600bps). Default value is 5.
parity	The type of parity checking, if any, supported by the service provider. Possible values are 0 (none), 1 (odd), 2 (even), 3 (mark), and 4 (space). Default value is 2.

Parameter	Purpose
<code>databits</code>	The number of data bits supported by the service provider. Possible values are 2 (7 bits) and 3 (8 bits). Default value is 2.
<code>stopbits</code>	The number of end-of-byte bits supported by the service provider. Possible values are 1, 1.5, and 2. Default value is 1.
<code>flowcontrol</code>	The type of handshaking supported by the service provider to prevent data loss during transmission. Possible values are 0 (none), 1 (XONXOFF), 2 (CTSRTS), and 3 (DSRDTR). Default value is 2.

The `alpha-pager` parent element also includes a `sender id`, which identifies the modem that is used to communicate with the specified paging central location(s), as well as one or more `device priority` child elements that specify what port is used.

Note that it is typical to have several `<paging-central>` definitions since your staff might have pagers (cell phones) from different vendors, and each vendor has their own phone number for paging. While creating action profiles, the vendor is specified using the `pager-pin@pager-central-name` syntax.

If the modem is not available or busy, pages are queued on the DGE. Undeliverable pages older than 1 hour are ignored (these parameters can be controlled via the configuration in `netvigil.xml` also).

Sample `netvigil.xml` 'paging central' configuration

```

<alpha-pager>
  <sender id="3035557777"/>
  <device priority="10" port="/dev/ttyS0" />
  <device priority="20" port="/dev/ttyS2" />
  <paging-central name="attws">
    <!-- name should be unique -->
    <number>9998887777</number> <!-- number to dial, including prefix -->
    <speed>5</speed> <!-- 0=110bps, 2=300bps, 4=1200bps -->
    <!-- 5=2400bps, 6=4800bps, 7=9600bps -->
    <parity>2</parity> <!-- 0=none, 1=odd, 2=even, 3=mark -->
    <!-- 4=space -->
    <databits>2</databits> <!-- 2=7bits, 3=8bits -->
    <stopbits>1</stopbits>
    <flowcontrol>1</flowcontrol> <!-- 0=none, 1=xonxoff, 2=ctsrts -->
  </paging-central>
</alpha-pager>

```

```

                                <!-- 2=ctsdrtr, 3=dsrdtr          -->
</paging-central>
<paging-central name="nextel">  <!-- name should be unique      -->
  <number>3035551212</number>  <!-- number to dial, including prefix -->
  <speed>5</speed>             <!-- 0=110bps, 2=300bps, 4=1200bps  -->
                                <!-- 5=2400bps, 6=4800bps, 7=9600bps -->
  <parity>0</parity>           <!-- 0=none, 1=odd, 2=even, 3=mark  -->
                                <!-- 4=space                      -->
  <databits>3</databits>       <!-- 2=7bits, 3=8bits          -->
  <stopbits>1</stopbits>
  <flowcontrol>0</flowcontrol> <!-- 0=none, 1=xonxoff, 2=ctsrts  -->
                                <!-- 3=dsrdtr                      -->
</paging-central>
</alpha-pager>

```



4.4 Customizing the Notification Content

The notification content for the built in notifications can be customized by editing the following files in the `NETVIGIL/etc/actions/` directory:

```

regular-email.xml
compact-email.xml
tap-pager.xml

```

There can be two sections in each file, one for the test threshold violations (`type="test"`) and one for the traps and log messages (`type="message"`). All the variables that are used in the plugin framework (see Section 26.2, "Creating New Plugin Actions" on page 331) are available for these notification xml files as well.

All multiline text in the `<body>` parameter is combined into a single line. Any `"\r\n"` or `"\n"` strings are converted into a newline character, unless it is prefixed by a `'^'` character. Hence, the following config:

```

<property name="body">
a\r\n
  b
  c   ^ d   \n

```

is converted into:

```

a
bc  d

```

4.5 Smart Suppression (Alarm Floods)

The actions and notification module takes network topology and other rules into account while triggering a notification to avoid alarm floods. When an upstream device fails, all downstream devices are unreachable and notifications can be suppressed. Furthermore, if “smart suppression” is enabled, then notifications for an application being unreachable because a server fails can also be suppressed.

4.6 Extending the Action Framework

The action framework can be extended easily using the Plug-in Framework (see Chapter 26, “Plugin Actions”) to run any external program. The device name and test information can be passed to the external program to build very flexible actions (which can then use the API to query the state of another device and test before executing a corrective action).

4.7 Remedy ARS Plugin

Remedy ARS is a commercially available trouble ticketing and CRM system. The Remedy plugin for NetVigil allows opening a new trouble ticket in Remedy ARS automatically as an ‘action’ when a test goes into any desired state. To avoid opening spurious tickets, you can delay the opening of a ticket until the test has failed several times using the flexible Action framework in NetVigil.

Note: This plugin needs to be licensed separately, please contact Fidelia for more information.

This integration package adds a new custom action to the drop-down list of actions available to a user of the NetVigil web application. The action can be configured to trigger after certain number of test cycles, repeat after several test cycles, and trigger during certain hours of the day, like any NetVigil action. Once triggered, the script connects to an existing Remedy (version 4 or 5) system and searches for a ticket in the specified queue matching certain subject (created using device and test name). If found, the ticket is updated with new information. Otherwise a new ticket is created and the URL to the ticket is added to the device comment.

4.7.1 Pre-requisites

Before installing this package, the following tasks need to be completed:

1. If you have multiple **locations** defined in your NetVigil environment, decide which locations should have the ability to open tickets in Remedy. As each location may have multiple DGE, this will assist you in compiling a list of DGE where the package will need to be installed.
2. This package uses the ARS Perl API and requires the ARS Perl modules to be functional. In order for this tool to function properly, ARS Perl must be installed and configured properly on each DGE (if the web application is running on a separate host, there is no need to install ARS Perl on that host).
3. Create a new login for NetVigil in Remedy. Set an appropriate username (e.g. netvigil) with a password. The password may not be blank.
4. Make sure the newly created user has permissions to create new ticket and add comments to existing tickets
5. Make sure the ARS Perl modules are working properly. The `test-remedy.pl` test script should be able to search and display all new/open tickets in your Remedy system (replace `$opt_XXX` parameters in the script with valid values). Create and run the script from each DGE to verify proper installation and communication with Remedy.

4.7.2 Installation

1. Copy the integration package to each DGE (that should have the custom action, and also to the host running the web application. Store it in a temporary location (e.g. /tmp)
2. Extract the files and start installation:

```
cd /tmp
gunzip -c integ-remedy-n.n.tar.gz | tar xvf -
cd integ-remedy-n.n
perl ./install.pl
```
3. Provide answers to the requested questions. The installation process will copy the integration package into appropriate location under NetVigil installation directory

4. You will need to restart the DGE process and web application at a convenient time before the action will be visible in the drop down list (in web application), or can be executed by a DGE

4.7.3 Configuration

1. To use the newly added plugin action, you first need to create an action profile that uses this script. Create an action profile via *Admin* -> *Actions* -> *Create New Action* (or update an existing profile). From the *Notify Using* drop-down list, you should be able to select the script. The name displayed on the list will correspond to the `<name>...</name>` parameter in `$NETVIGIL_HOME/plugin/actions/createTicketInRemedy.xml` file. The *Message Recipient* field can be left empty and rest of the parameters set as you see fit. Now apply this action profile to various tests as required.
2. By default, a new ticket will be created on a per-test basis. If device A has two tests X and Y, and both tests fail, one ticket for X and one ticket for Y will be created. If you prefer to restrict new tickets to a per-device basis, where information for X and Y will be entered into same ticket (second test information will be added as additional comment), then edit the XML configuration file for the script and add `--perdevice` option to the `<parameters>..</parameters>` section

4.7.4 Troubleshooting

1. Look in `netvigil/logs/netvigil.error` for any error messages logged by the DGE process
2. Check to make sure `addon/actions/createTicketInRemedy.pl` is executable (mode 0555)
3. Try running the script manually to ensure you can create a new ticket:

```
cd /usr/local/netvigil/addon/actions
./createTicketInRemedy.pl --arsserver YOUR_REMEDY_SERVER \
--arsuser netvigil --arspasswd YOUR_PASSWORD \
--form HPD:Helpdesk
--device "Sample Device" --test "Sample Test" \
--severity warning --result 100 --unit ms \
--location "Data Center" --type ping/rtt \
--threshold "75/200" --search
```

4.8 RT Trouble Ticketing Plugin

This integration package adds a new custom action to the drop-down list of actions available to a user of the NetVigil web application. The action can be configured to trigger after certain number of test cycles, repeat after several test cycles, and trigger during certain hours of the day, like any NetVigil action. Once triggered, the script connects to an existing RT (version 2.x) system and searches for a ticket in the specified queue matching certain subject (created using device and test name). If found, the ticket is updated with new information. Otherwise a new ticket is created and the URL to the ticket is added to the device comment.

Note: This plugin needs to be licensed separately, please contact Fidelia for more information.

4.8.1 Pre-requisites

Before installing this package, the following tasks need to be completed:

1. If you have multiple **locations** defined in your NetVigil environment, decide which locations should have the ability to open tickets in RT. As each location may have multiple DGE, this will assist you in compiling a list of DGE where the package will need to be installed.
2. This package uses RT Perl API and requires the RT Perl modules to be functional. In order for this tool to function properly, RT must be installed and configured properly on each DGE (if the web application is running on a separate host, there is no need to install RT on that host). Install RT under its default location `/opt/rt2`, or install it at a location of your choice, and create a symbolic link from `/opt/rt2` to that directory. Instructions for installing RT are available from www.bestpractical.com/rt
3. Copy `etc/config.pm` from your RT host to `/opt/rt2/etc/config.pm` on all the DGE. Edit `/opt/rt2/etc/config.pm` and update `$DatabaseHost` to point to your RT host. You also may want to update `$LogDir`, or create the directory specified and make sure that the directory permissions are setup properly

4. Create a new login for NetVigil into RT database (via WebRT). Set an appropriate username (e.g. netvigil@your.domain) but make sure to leave the email field blank. This will make sure that when a new ticket is created, no auto-replies will be sent (if there is such a **script** configured for the queue you will be using)
5. Make sure the newly created user has permissions to create new ticket and add comments to existing tickets
6. You may have to configure your RT database for remote access. By default, when using MySQL for RT database, the database user (specified in `/opt/rt2/etc/config.pm`, variable `$DatabaseUser` is only allowed access from localhost. Before this custom action can create/update tickets, it will need to be allowed access. For MySQL, this involves connecting to the **rt2** database (or the database name specified in `/opt/rt2/etc/config.pm`) locally on the RT host as root, and using:

```
GRANT SELECT, INSERT, CREATE, INDEX, UPDATE, DELETE ON rt2.* TO
rt_user@n.n.n.n;
```

where `n.n.n.n` is the IP address of each DGE. When using Postgres database, you will have to make necessary additions to `data/pg_hba.conf` file. Please refer to configuration documents for the respective database vendors (MySQL or PostgreSQL) for additional details.

7. Make sure the RT Perl modules are working properly. The `test-rt.pl` test script should be able to search and display all new/open tickets in your RT system (replace `YOUR_QUEUE_NAME` in the script with a valid queue name). Create and run the script from each DGE to verify proper installation and communication with RT.

4.8.2 Installation

1. Copy the integration package to each DGE (that should have the custom action, and also to the host running the web application. Store it in a temporary location (eg. `/tmp`)
2. Extract the files and start installation:

```
cd /tmp
gunzip -c integ-rt2-n.n.tar.gz | tar xvf -
cd integ-rt2-n.n
perl ./install.pl
```

3. Provide answers to the requested questions. The installation process will copy the integration package into appropriate location under NetVigil installation directory
4. You will need to restart the DGE process and web application at a convenient time before the action will be visible in the drop down list (in web application), or can be executed by a DGE

4.8.3 Configuration

1. To use the newly added plugin action, you first need to create an action profile that uses this script. Create an action profile via *Admin* -> *Actions* -> *Create New Action* (or update an existing profile). From the *Notify Using* drop-down list, you should be able to select the script. The name displayed on the list will correspond to the `<name>...</name>` parameter in `$NETVIGIL_HOME/plugin/actions/createTicketInRT.xml` file. The *Message Recipient* field can be left empty and rest of the parameters set as you see fit. Now apply this action profile to various tests as required.
2. If you wish to create tickets in different queues, you will need to create two different plugin actions. one each for the two RT queue:

```
su
cd /path/to/netvigil
etc/netvigil.init stop
cd plugin/actions
mv createTicketInRT.xml RT-queue1.xml
cp RT-queue1.xml RT-queue2.xml
```

Now edit `RT-queue1.xml` and change the `"<name>.. </name>"` option to something descriptive, like

```
<name>Create/Update RT-queue1</name>
```

Also update the `--queue` option to **queue1**. Save the file and make similar changes for `RT-queue2.xml`. Make sure to use different `"<name>..</name>"` options

3. By default, a new ticket will be created on a per-test basis. If device A has two tests X and Y, and both tests fail, one ticket for X and one ticket for Y will be created. If you prefer to restrict new tickets to a per-device basis, where information for X and Y will be entered into

same ticket (second test information will be added as additional comment), then edit the XML configuration file for the script and add `--perdevice` option to the `<parameters>..</parameters>` section

4.8.4 Troubleshooting

1. Look in `netvigil/logs/netvigil.error` for any error messages logged by the DGE process
2. Check to make sure `plugin/actions/createTicketInRT.pl` is executable (mode 0555)
3. Try running the script manually to ensure you can create a new ticket:

```
cd /usr/local/netvigil/plugin/actions
./createTicketInRT.pl --queue YOUR_QUEUE_NAME \
--rtuser netvigil@your.domain \
--device "Sample Device" --test "Sample Test" \
--severity warning --result 100 --unit ms \
--location "Data Center" --type ping/rtt \
--threshold "75/200" --search
```

