

Advanced NetVigil Topics



12.1 External Help

External help provides NetVigil operators the ability to write support documentation specific to a Department, device or test and tie it directly to that same object via a **HELP** link in the Web UI. This way, less experienced system administrators can be provided with a first line of troubleshooting in the absence of live support. You can also enable actions (e.g., server restart) via the **HELP** links. This is a powerful option, as any number of files can be configured to work in this fashion, enabling a large number of background processes via the web app.

The perl script `displayTestHelp.pl` in the `utils` directory scans through `NETVIGIL_HOME/plugin/help` for help text specific to a Department, device or test. This script expects one argument in the form:

```
<department_name> | <device_name> | <device_addr> |  
<test_type> | <test_subtype> | <test_name>
```

where `device_addr` can be fqdn or ip address. This has to match what was used for device creation. The field `test_name` should match the descriptive name that was displayed during test creation (or in test details page).

NOTE *The perl script converts everything (i.e. `acct_name` & `device_name`) to lowercase to avoid any case related problems when searching for the file. Therefore, the directories and subdirectories must be named in lowercase and formatted the same (i.e. spaces or special chars included) as the Department and device names.*

The script searches `NETVIGIL_HOME/plugin/help` according to following algorithm:

1. Search for directory - `acct_name` ELSE `_default_user`, if found, cd into it.

2. Search for subdirectory - device_name ELSE device_addr ELSE _default_device, if found, cd into it.
3. Search for the files in the current directory in the following order:
`<test_type>_<test_subtype>_<test_name>.{html,txt}` ELSE
`<test_type>_<test_subtype>.{html,txt}` ELSE
`<test_type>.{html,txt}` ELSE
`default.{html,txt}`
4. Display the entire file on stdout (if text, then put HTML tags around the text).
5. If not found, display NO FILE FOUND on stdout in HTML format. The script prints out errors on stdout. The location of the script is specified in web.xml and it can basically be any script (or program). It is up to the target script to take the arguments and send back help text in the required format.

For example, to create a help file for device 'mail_server' and a more specific one for the 'disk_space', in Department 'local_department':

```
cd NETVIGIL_HOME/plugin/help
mkdir -p local_department/mail_server
mkdir -p local_department/_default_device
cd local_department/
vi _default_device/default_html
vi mail_server/snmp_disk.txt
vi mail_server/default.html
```

It is possible to use your own script, that (for example) connects to a database and retrieves escalation information based on specified criteria.

12.2 High Availability Configurations

The NetVigil distributed database and processing architecture allows very high levels of fault tolerance and scalability during deployment. All of the components in the various tiers are horizontally scalable which is essential for expansion and real-time performance reports.

All of the configuration information is stored in the BVE configuration database. On startup, the DGEs connect to the BVE configuration database and download a local copy of their configuration. Any updates made to the BVE configuration database are pushed out in real-time to the corresponding DGE.

To handle the case of a DGE physical server going down, you can setup a spare 'hot standby' server in any central location (N+1 redundancy) which has the software installed and configured. In the case of a production DGE going down for an extended period of time due to hardware failure, you can set the name of the DGE in the `dge.xml` config file (see "DGE Identity" on page 34) and start NetVigil on the backup server. This backup DGE will automatically connect to the BVE configuration database and download the configuration of the failed DGE. When the production DGE comes back up, it can be even run in parallel before shutting down the backup DGE. The only caveat is that the performance data collected during this interval will be missing on the production DGE (this requirement of synchronizing the performance data will be addressed in a future release).

If desired, you can have a backup DGE for each of the production DGEs (N+N redundancy) but this is not really needed if the centralized DGE can poll all the data remotely.

If connectivity between the DGE and the BVE database is lost, the DGE continues to poll, aggregate and even generate alarms completely independently. When connectivity to the BVE database is restored, the DGE restarts and downloads a fresh copy of its configuration database.

The BVE database can be replicated on multiple servers for fault tolerance. In the future, the DGEs will be able to automatically fail-over to an alternate BVE database if the primary database is not reachable.

The performance database which is local to each DGE can be located on a remote database cluster if needed for fault tolerance also. The JDBC communication between the DGE and the performance database allows such a setup seamlessly just by a few configuration file changes.

Lastly, the Web application and reporting engine also gets all the configuration information from the BVE database server on startup and hence you can have any number of web application servers behind a load balancer for fault tolerance as well as distributed report processing.

12.3 Customizing NetVigil

The NetVigil UI is web based, and can be updated by editing HTML stylesheets. Additionally, different languages can be supported by setting the locale and creating a properties file.

Changing Skin

To customize the look and feel (skin) of NetVigil, you can create a new stylesheet and icons under `tomcat/webapps/ROOT/resource/skins/`

Changing Languages

You can override the default NetVigil resource file by adding your own locale specific resource file. As an example, to add support for French, you will need to create the following file:

```
plugin/locale/EmeraldResources_fr.properties
```

Then edit `etc/webapp.init` and add the location of this file to the `CLASSPATH`:

```
CLASSPATH=${INSTALL_DIR}/plugin/locale:${CATALINA_CP}:${CLASSPATH}
```

Now restart the web application and select your Locale from `Manage>Prefs`.