HP Network Node Manager i Software

For the Windows®, HP-UX, Linux, and Solaris operating systems

Software Version: 9.22

Reference Pages

Document Release Date: November 2012
Software Release Date: November 2012

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe® is a trademark of Adobe Systems Incorporated.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

Oracle Technology — Notice of Restricted Rights

Programs delivered subject to the DOD FAR Supplement are 'commercial computer software' and use, duplication, and disclosure of the programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, programs delivered subject to the Federal Acquisition Regulations are 'restricted computer software' and use, duplication, and disclosure of the programs, including documentation, shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software-Restricted Rights (June 1987). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

For the full Oracle license text, see the license-agreements directory on the NNMi product DVD.

Acknowledgements

This product includes software developed by the Apache Software Foundation. (http://www.apache.org)

This product includes software developed by the Indiana University Extreme! Lab. (http://www.extreme.indiana.edu)

Network Node Manager Reference Pages

User Commands (1)	Administrator Commands (1M)	File Formats (4)
nmscatgets nnmcluster nnm.envvars nnmprops nnmsetcmduserpw.ovpl nnmtrapdump.ovpl nnmfindattachedswport.ovpl nnmversion.ovpl ovjbosspath.ovpl ovjrepath.ovpl ovstatus nnmdumpevents nnmsnmpnotify.ovpl	nmsdbmgr nnmaction nnmbackup.ovpl nnmbackupembdb.ovpl nnmcertmerge.ovpl nnmchangedbpw.ovpl nnmchangeembdbpw.ovpl nnmchangesyspw.ovpl nnmcommconf.ovpl nnmcommload.ovpl nnmconfigexport.ovpl nnmconfigimport.ovpl nnmconfigpoll.ovpl nnmconnedit.ovpl nnmdeleteattributes.ovpl nnmdeleteurlaction.ovpl nnmdisableperfspi.ovpl nnmdiscocfg.ovpl nnmenableperfspi.ovpl nnmhealth.ovpl nnmicons.ovpl nnmincidentcfg.ovpl nnmincidentcfgload.ovpl nnmincidentcfgdump.ovpl nnmldap.ovpl nnmlicense.ovpl nnmloadinterfacegroups.ovpl nnmloadipmappings.ovpl nnmloadattributes.ovpl nnmloadmib.ovpl nnmloadnodegroups.ovpl nnmloadseeds.ovpl nnmmanagementmode.ovpl nnmnodedelete.ovpl nnmnodegroup.ovpl nnmnoderediscover.ovpl nnmofficialfqdn.ovpl nnmooflow.ovpl nnmopcexport.ovpl nnmperfspisync.ovpl nnmresetembdb.ovpl nnmrestore.ovpl nnmrestoreembdb.ovpl nnmsecurity.ovpl nnmseeddelete.ovpl nnmsetdampenedinterval.ovpl nnmsetiospeed.ovpl nnmsetofficialfqdn.ovpl	disco.NoVLANIndexing disco.SkipXdpProcessing hostnolookup.conf ipnolookup.conf macdedupexceptions.txt nnm.ports nnm.properties ldap.properties nnmtrapd.conf trapFilter.conf hosted-object-trapstorm.conf UnnumberedNodeGroup.conf UnnumberedSubnets.conf incidentconfiguration.format

	nnmsnmpbulk.ovpl nnmsnmpget.ovpl nnmsnmpnext.ovpl nnmsnmpset.ovpl nnmsnmpwalk.ovpl nnmsso.ovpl nnmstatuspoll.ovpl nnmtopodump.ovpl nnmtrapconfig.ovpl nnmtrapdload.ovpl nnmtrimincidents.ovpl nnmwhat ovaddobj ovdelobj ovjboss ovserror ovspmd ovstart ovstop pmd pmdmgr	

Name

nmscatgets — Get localized catalog strings for NNMi

SYNOPSIS

`nmscatgets` [-f *`CATFILE`* -s *`SETNUM`* -m *`MSGNUM`* -d *`DEFAULT`*]

DESCRIPTION

The `nmscatgets` command is used to query a message catalog and retrieve a localized (non-English) string from that catalog file. This string can then be displayed, for example, from a perl script. The caller must know the message catalog containing the desired localized strings, as well as the set-number and message-number within that file. If the specified message is not found, the default (English) value is returned.

The `nmscatgets` command is mostly used internally by other NNMi or HP-provided scripts. It doesn't provide much value for other uses.

Parameters

The `nmscatgets` command recognizes the following options.

`-f` *`FILE`*

The file name containing the localized strings to query.

`-s` *`SETNUM`*

The set number within the catalog file, generally "1".

`-m` *`MSGNUM`*

The message number within the catalog file.

`-d` *`DEFAULT`*

The default (English) string to be printed if one or more of the *`FILE`*, *`SETNUM`*, or *`MSGNUM`* values are not available.

RETURN VALUE

The `nmscatgets` command returns the localized string when the *`FILE`*, *`SETNUM`*, and *`MSGNUM`* values are all valid. Otherwise the `nmscatgets` command returns the DEFAULT (English) string provided to the command.

AUTHOR

`nmscatgets` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmcluster — start NNMi cluster services

SYNOPSIS

`nnmcluster` [-disable|-enable] [-display] [-interfaces] [-startnnm|-stopnnm] [-acquire|-relinquish] [-shutdown [-force]] [-dbsync] [-halt] [-node *nodename*] [[-daemon]]

DESCRIPTION

`nnmcluster` starts the NNMi cluster process. The NNMi cluster command permits an administrator to set up two systems for ensuring the availability of NNMi services if one system fails. After you run the `nnmcluster` command on each node, each one will detect the other and form a cluster. The first node to join the cluster comes up in the `active` state, and starts the NNMi services (using the `ovstart` command). The second node detects that there is already an active node, and assumes the `standby` state. If the standby node loses connectivity with the active node, (due to system shutdown or failure), then the standby node assumes the `active` state and starts the NNMi services.

If the `nnmcluster` is called with no command-line parameters, it starts the cluster in interactive mode. The interactive mode permits the system administrator to view and modify cluster settings in an interactive session. These settings include the ability to enable or disable automatic failover, shutdown a node in the cluster, transfer NNMi services from active to standby, and other settings.

If `nnmcluster` is called with the `-daemon` parameter, the NNMi cluster starts up as a background daemon process or Windows service.

If the `nnmcluster` command is called with other command-line parameters, it will initiate the actions specified on the command-line. These actions typically affect the NNMi cluster daemon process on the local node. However, if the `-node` *`nodename`* option is used, it affects the NNMi cluster daemon process on the specified `node`.

Most of the options available from the command-line are also available in interactive mode. For example, using the `-shutdown` option from the command line is the same as using the `shutdown` command in interactive mode. The interactive mode has some additional commands, such as `help`, to display a list of available commands, and `quit`, to exit the interactive mode. The `-node` *`nodename`* command is also available interactively.

Note that only NNMi cluster daemon processes are capable of starting NNMi services. The interactive mode and specifying actions on the command-line are methods for affecting the behavior of a daemon process on one of the nodes in the cluster. For example, using the `-acquire` option causes the daemon process on the local node (or the specified node if used with the `-node` option), to acquire the `active` state and start NNMi services. After an NNMi cluster daemon process is started, the only way of interacting with that daemon process is by using the command line or interactive mode settings. For example, if you want to terminate that daemon process, use the `nnmcluster -shutdown` command.

When NNMi is using the embedded database, the NNMi cluster application synchronizes the database between the active and standby nodes. This is achieved by sending a complete database backup to the

standby, followed by periodic incremental database transaction logs. The time intervals for the frequency of full backups and transaction logs, along with other cluster parameters, are defined in the `nms-cluster.properties` file.

For the embedded database scenario, the NNMi cluster application has a startup period that must occur before permitting the active node to send the database to the standby node. During this startup period, command options that would transfer the `active` state to the standby node are disabled. These options include `shutdown`, `acquire`, `relinquish`, and other available options, as they can leave the standby node in a state where it only has a portions of the database and is unable to run NNMi. However, after the standby node has received the entire database, there should be no critical times from that point forward, assuming both systems stay running (a restart causes the standby node to verify that it is still in-sync with the active node).

If NNMi is using the Oracle database, then no database replication is performed, as the Oracle database instance is on a third machine outside of NNMi control. In this scenario, the NNMi `nnmcluster` command still monitors the active NNMi management server and starts NNMi on the standby server if the active fails.

Parameters

The `nnmcluster` command recognizes the following options. Any unrecognized options are reported by a usage message. Options are always processed in the specified order. For example, using `-display -disable` is different from using `-disable -display`. All of the parameters refer to the NNMi cluster daemon process on the local node, unless the `-node` *`nodename`* option is used.

`-disable`

Disables the automatic failover capability (automatic failover is enabled by default). The system administrator might choose to shutdown the active node briefly to perform some administrative tasks. The `-disable` parameter permits the active node to be shut down without the standby node becoming active and starting the NNMi services. You can restart the same node in the `active` state by using the `-acquire` option followed by the `-enable` option.

`-enable`

Re-enable automatic failover capability after being disabled as described above.

`-interfaces`

List the network interfaces (NICs) on the system, displaying the system nomenclature and the Java nomenclature. On Unix platforms, these are the same value, e.g. "eth0", "lan1", "bge3", etc. On Windows, the names are different, e.g. "Network Interface 1" might map to "eth3". The purpose of this parameter is to control the NIC used for NNMi cluster communications; e.g. to choose a management NIC instead of the data NIC. The NNMi cluster needs to know the Java name, e.g. "eth3" in the Windows case listed above.

`-display`

Connect to the cluster, query the current cluster state, then display that state to the administrator.

`-startnnm`

The active node may not be running NNMi services. For example, suppose you stopped the NNMi services earlier using the `-stopnnm` option. You can start up the NNMi services on the active node by using the `-startnnm` option.

`-stopnnm`

Shut down the NNMi services on the active node, but do not release the `active` state. If you use this option, NNMi does not generate a failover event. For example, the standby node does not assume the `active` state.

`-acquire`

The system administrator might want to transfer NNMi services from a node which is currently in the `active` state to another node which is currently in the `standby` state. Without using the `-node` option, the local system becomes the new active node. If the administrator uses the `-node` *`nodename`* option, the specified node becomes the new active node.

`-relinquish`

Permits the system administrator to give up the `active` state on the local (currently active) node and causes the NNMi services to transfer control to the current standby node. The node being relinquished stops NNMi services and assumes the `standby` state.

`-dbsync`

Permits the system administrator to trigger an immediate database synchronization on active node. This option is only applicable when NNMi is using the embedded database (not using an Oracle database).

`-shutdown [-force]`

Shuts down the NNMi cluster daemon process on the local node. If the `-node` option is provided, then shut down the NNMi cluster daemon process on the specified node. The NNMi cluster program tries to prevent you from shutting down the cluster during times when doing so might compromise the data integrity on the standby server. For example, if the standby server is receiving a full database backup from the active server, it would be bad to have a failover event during that time. If you try to shut down the NNMi daemon process, you might get a message stating that the cluster is in a transition state, as in the standby node is receiving important data, so you need to permit it to complete before attempting ot shut down the cluster daemon process. If you want to force a shutdown anyway, use the "`-force`" option. This disables failover to the standby (since it is in an incomplete state), then shuts down the local (or specified) node.

`-halt`

Shuts down the NNMi cluster daemon process on all nodes in the cluster. This disables failover, shuts down all standby nodes, then shuts down the active node.

`-node` *`nodename`*

Causes one or more actions specified on the command line to affect the NNMi cluster daemon process on the specified `nodename`. Without specifying this option, the actions specified on the command line to apply to the local NNMi cluster daemon.

`-daemon`

Starts the NNMi cluster as a daemon. The command is immediately launched in the background. No other command line parameters can be specified in daemon mode.

RETURN VALUE

If running the `nnmcluster` command with command line options results in success, the command exits with the `0` (zero) status (there are no errors). However, if running the `nnmcluster` command with command line options results in failure, the command exits with the `1` status (there are errors). In interactive mode, the exit status is always 0.

In daemon mode the `nnmcluster` command launches as a background process and the shell prompt is immediately returned. After you start a daemon process, you can monitor cluster status by using the `nnmcluster -display` or `ovstatus` commands: the NNMi cluster determines if or when to start NNMi services, depending on the `active` or `standby` status of the node in relation to other nodes in the cluster; `ovstatus` displays a `not running` status on the standby node, but running the `nnmcluster -display` command tells you that the node is in the `standby` state.

DIAGNOSTICS

`nnmcluster` logs output to the NNMi log directory (`/var/opt/OV/log/nnm` on Unix, and `%NnmDataDir%\log\nnm` on Windows). Each actively-running instance of the `nnmcluster` process, such as a daemon running in parallel with either or both the interactive or command-line mode, has a separate log file. The most-recent running threads will always be `nnmcluster.0.*.log`.

The NNMi cluster internally uses an open-source technology called *JGroups*. The log files for JGroups are in the same directory shown above, and are named `jgroups.log`.

EXAMPLES

```
nnmcluster -daemon
```

```
nnmcluster -display
```

The first command starts the NNMi cluster as a daemon process, and immediately returns a shell prompt, leaving a background daemon process. You can query, shut down, or apply other actions to this daemon process using interactive or command-line modes. You can repeatedly use the second command to monitor the status of the cluster, specifically the local daemon process, to determine if it comes up in the `active` or `standby` state.

```
nnmcluster -shutdown -node xyz.mycompany.com
```

Causes the NNMi cluster daemon process on the specified node to shut down. If that node is the current active node, and if automatic failover is enabled, then the NNMi services will transfer to the standby node.

```
nnmcluster
```

Enter the interactive mode of the NNMi cluster command to view or modify cluster parameters. To exit this program, use the `exit` or `quit` commands.

```
nnmcluster -acquire
```

Cause the daemon-mode NNMi cluster process on the local system to become the active node. The current active is set to standby mode then NNMi services start on the local node.

The following is a common system administration sequence to permit the administrator to temporarily shut down the NNMi cluster on the active node, then bring it back up later as active, without causing a failover event to the standby node.

Step 1: Run the **nnmcluster -disable -shutdown** command.

Step 2: Perform some system administration tasks.

Step 3: Run the **nnmcluster -daemon** command.

Step 4: Run the **nnmcluster -display** command. Use this command to determine when the daemon is up.

Step 5: Run the **nnmcluster -enable** command.

The command shown in step 1 disables failover first, then shuts down the local daemon process. In step 2, the system administrator performs some administration tasks without the risk of triggering a failover. The command shown in step 3 restarts the daemon NNMi cluster process. You can repeatedly use the command shown in step 4, to determine when the local daemon process is up and NNMi is running. The command shown in step 5 re-enables automatic failover after NNMi is running on the active node.

AUTHOR

`nnmcluster` was developed by Hewlett-Packard Company.

FILES

Windows: `$%NNM_PROPS%\nms-cluster.properties`

UNIX: `$NNM_PROPS/nms-cluster.properties`

This file defines the cluster parameters; specifically, you must uniquely name the cluster to differentiate it from other NNMi clusters that might be on the same network. You can optionally set other parameters such as timeouts.

SEE ALSO

ovstart(1M), ovstop(1M).

Return to Reference Pages Index

Name

nnm.envvars — A script to define the environment variables for universal paths in NNMi.

SYNOPSIS

Windows operating systems:

`nnm.envvars.bat`

UNIX operating systems:

`nnm.envvars.sh`

`nnm.envvars.csh`

DESCRIPTION

The `nnm.envvars` script defines the NNMi environment variables for universal paths. Universal paths simplify the use of NNMi and other HP Software products by providing paths and filenames common to all operating system platforms. Universal paths are provided for the Windows command interpreter and UNIX shells.

On UNIX operating systems, you can modify the `.profile` or the `.login` file, so that the shell script is sourced (activated) each time you log on to your system. Alternatively, you can activate the file for an individual terminal, user, and session by sourcing the file into the current environment.

To view the universal paths that are defined, read the `nnm.envvars.sh` file, which resides in `/opt/OV/bin` on UNIX operating systems or `%NnmInstallDir%\bin` on Windows operating systems.

EXAMPLES

To modify the `.profile` or `.login` file, add the appropriate line from the following examples to the file.

To source the `nnm.envvars` script, use one of the following commands.

- From a Windows command line:

  `%NnmInstallDir%\bin\nnm.envvars.bat`

  Where: `%NnmInstallDir%` is the directory where NNMi is installed. Note that the NNMi installer creates this variable as a system environment variable.

- From a Windows operating system with UNIX-style shells installed (for example, MKS Toolkit or CygWin):

  Using `sh`, `ksh`, or `bash`:

  `$NnmInstallDir/bin/nnm.envvars.sh`

Where: `$NnmInstallDir` is the directory where NNMi is installed.

Using `csh`:

```
source $NnmInstallDir/bin/nnm.envvars.csh
```

Where: `$NnmInstallDir` is the directory where NNMi is installed.

- From UNIX operating systems:

  Using `sh`, `ksh`, or `bash`:

  ```
  . /opt/OV/bin/nnm.envvars.sh
  ```

  Using `csh`:

  ```
  source /opt/OV/bin/nnm.envvars.csh
  ```

AUTHOR

`nnm.envvars` was developed by Hewlett-Packard Company.

FILES

Windows operating systems:

`%NnmInstallDir%``\bin\nnm.envvars.bat` (for Windows command line)

`%NnmInstallDir%``\bin\nnm.envvars.sh` (for sh, ksh, or bash)

`%NnmInstallDir%``\bin\nnm.envvars.csh` (for csh)

UNIX operating systems:

`/opt/OV/bin/nnm.envvars.sh` (for sh, ksh, or bash)

`/opt/OV/bin/nnm.envvars.csh` (for csh)

EXTERNAL INFLUENCES

International Code Set Support: Supports single-byte and multi-byte character code sets.

Return to Reference Pages Index

Name

nnmprops — Query values of NNMi properties.

SYNOPSIS

`nnmprops` [-l] [-q *`prop`*] [-m *`match`*] [-e *`expand`*]

DESCRIPTION

`nnmprops` is used to query the property values that run NNMi processes. The `nnmprops` command consolidates and displays these properties, which are stored in several locations in the file system. This command can be used in other scripts that need to query and act on the values of NNMi system properties.

Parameters

`nnmprops` recognizes the following options.

`-l`

Lists the value of all properties.

`-q` *`PROP`*

Queries the specified property. This option can be repeated to query multiple properties.

`-m` *`STRING`*

Queries all properties that start with the prefix STRING.

`-e` *`STRING`*

Expand any properties in the STRING with their corresponding value.

RETURN VALUE

`nnmprops` always exits with the status `0` (zero) if no errors were encountered; `1` otherwise.

EXAMPLES

```
nnmprops -l
```

Lists all properties and their values.

```
nnmprops -q com.hp.nms.ui.sso.isEnabled -q com.hp.nms.ui.sso.domain
```

Queries the property values of the *com.hp.nms.ui.sso.isEnabled* and *com.hp.nms.ui.sso.domain* properties.

```
nnmprops -m com.hp.nms.ui.sso
```

Queries all properties that start with `com.hp.nms.ui.sso`. For example, the values for `.protectedDomains`, `.domain`, `.initString`, and `.isEnabled` would be returned.

```
nnmprops -e "The values for com.hp.nms.ui.sso are ${com.hp.nms.ui.sso}."
```

Displays the string with the value of `${com.hp.nms.ui.sso}` expanded. Note that the "${" *and* "}" are required to delineate the property name.

AUTHOR

`nnmprops` was developed by Hewlett-Packard Company.

FILES

There are several properties files used by the nnmprops program. These files are separated into two categories: HP-provided default (out-of-the-box) values and user-modified override values. The intention is that HP can change the default properties values in future NNMi releases; however, user-modified values always override the HP-provided default.

The files found in the directory hierarchy below `%NnmInstallDir%\misc\nnm\props` (Windows) or `$NnmInstallDir/misc/nnm/props` (UNIX) define the HP-provided default values.

> **Note**
>
> You should never modify any file below this location since future NNMi versions could overwrite any modifications.

The files in the `%NnmDataDir%\shared\nnm\conf\props` (Windows), `%NnmDataDir%\conf\nnm\props` (Windows), `$NnmDataDir/shared/nnm/conf/props` (UNIX), and `$NnmDataDir/conf/nnm/props` (UNIX) define modified values or values changed programmatically at installation or runtime. These values are initially commented-out copies of the HP-provided values. To edit these files, remove the comment and change a value. The new value overrides the default value.

The difference between the two directories is as follows:

- `%NnmDataDir%\shared\nnm\conf\props` (Windows) and `$NnmDataDir/shared/nnm/conf/props` (UNIX) contain properties that are shared in a cluster (HA cluster or NNMi Application Failover cluster, for example).
- `%NnmDataDir%\shared\nnm\conf\props` (Windows) and `$NnmDataDir/shared/nnm/conf/props` (UNIX) defines values which are not shared. For instance, each node in the cluster may have different values for the same property.

Return to Reference Pages Index

Name

nnmsetcmduserpw.ovpl — set the account credentials to be used in place of the -u/-p options when executing scripts as this user.

SYNOPSIS

```
nnmsetcmduserpw.ovpl
```

DESCRIPTION

You can use `nnmsetcmduserpw.ovpl` to set the NNM account credentials to be used when executing any scripts normally requiring the -u/-p option. The user and password values will be used when scripts are executed by this user without the -u/-p command line arguments. NOTE: The UNIX user root should never run this command. The "root" user does not need to provide -u/-p by default.

To run this command, make sure you are logged into the system as the user that will be executing the command line scripts. These values are set on a per user basis.

Parameters

No supported parameters.

EXAMPLES

You may want to set up account username and password values to be used when executing command line scripts normally requiring the -u/-p option. This can be done to avoid specifying password information on the command line.

Usage is:

```
# nnmsetcmduserpw.ovpl

WARNING: This change will affect the credentials to be used in place
         of the -u/-p command line options whenever this user executes
         a script requiring these.  Please ensure you are logged in as
         the desired user before executing this script.
         Executing this script will create/edit the .nnm/nnm.properties
         file in the users home directory.
Would you like to continue? [n] y

Thank you!

Please provide the user name for the account.
user: myuser
Please provide a password for the user account.
password: mypass
enter password again: mypass

User/Password values stored successfully in /home/user/.nnm/nnm.properties
```

AUTHOR

`nnmsetcmduserpw.ovpl` was developed by Hewlett-Packard Company.

FILES

`nnmsetcmduserpw.ovpl` resides in `$NNM_BIN directory`.

SEE ALSO

nnm.properties(4).

Return to Reference Pages Index

Name

nnmtrapdump.ovpl — Print traps logged in the binary trap store to the console

SYNOPSIS

`nnmtrapdump.ovpl` [-u *`user`*] [-p *`password`*] [-t] [-from *`date`*] [-to *`date`*] [-source *`IP address`*] [-trapid *`Trap OID`*] [-last *`minutes`*] [-short] [-nodns] [-hexDump]

DESCRIPTION

All incoming traps are logged in a binary trap store by the HP NNM Trap Service. `nnmtrapdump.ovpl` can be used to see the traps that have been logged. It can also be used to watch new incoming traps. When using trap tool any log or error messages are printed to the standard error. By redirecting the standard error to a different file, you can prevent these messages from getting mixed with the actual trap dump output.

Parameters

`nnmtrapdump.ovpl` supports the following parameters:

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-t`

Use *`-t`* to continuously print incoming traps. This option cannot be used with the `-to` option

`-from` *`date`*

Specifies the date from which traps are printed. The date is specified in ISO 8601 standard format: *`yyyy-mm-ddThh:mm:ss[+ or -]hh:mm`*. This option can be used to drill down from the analytics entries in the trapanalytics.0.0.log file. This option cannot be used with the `-last` option

`-to` *`date`*

Specifies the date till which traps are printed. The date is specified in ISO 8601 standard format: *`yyyy-mm-ddThh:mm:ss[+ or -]hh:mm`*. This option can be used to drill down from the analytics entries in the trapanalytics.0.0.log file.

`-source` *`IP address`*

Specifies the source IP address of the traps that should be printed. Only traps from the given source will be printed.

`-trapid` *`Trap OID`*

Specifies the trap OID of the traps that should be printed. Only traps with the given OID will be printed.

`-last` *`minutes`*

Specifies the age of the traps to be printed. The value is in minutes. Only traps are at the most this old will be printed. This option cannot be used along with the `-from` option.

`-short`

Use *`-short`* to print a short format of incoming traps. Only the trap OID, arrival time and source address are printed

`-nodns`

Use *`-nodns`* to prevent resolution of IP addresses to node names. This speeds up the printing of traps

`-hexDump`

Use *`-hexDump`* to print traps in hexadecimal format

EXAMPLES

To print all the traps in the binary trap store to the console:

```
nnmtrapdump.ovpl
```

To wait in a loop and print all incoming traps:

```
nnmtrapdump.ovpl -t
```

To print traps within the 5 minute window from 9:00AM to 9:05AM MDT on July 31 2008:

```
nnmtrapdump.ovpl -from 2008-07-31T09:00:00-06:00 -to 2008-07-31T09:05:00-06:00
```

To print traps that came within the last 5 minutes and then wait for incoming traps:

```
nnmtrapdump.ovpl -last 5 -t
```

To print traps from IP address 192.168.0.1:

```
nnmtrapdump.ovpl -source 192.168.0.1
```

FILES

`$NNM_DB/traps` is the directory that contains the files that constitute the trap database.

`$NNM_LOG/trapanalytics.0.0.log` is the analytics log file containing information about the most frequent trapids and sources sending traps.

AUTHOR

`nnmtrapdump.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmfindattachedswport.ovpl — Find the switch port to which the input end node is attached

SYNOPSIS

`nnmfindattachedswport.ovpl` [-u <*user*>] [-p <*password*>] { -i <*end node file*> | -n <*end node*> } [-o <*output file*>]

DESCRIPTION

The **nnmfindattachedswport.ovpl** script displays the switch port that is connnected to an end node. When using the **nnmfindattachedswport.ovpl** script, specify the end node as a MAC address, an IP address, or a hostname. Specify the MAC address in upper case with no leading `0x` or `0X`. You can also specify the input as a seed file, using one line per entry within the seed file.

The display consists of the following:

- The end node.
- The switch hostname.
- The interface name of the switch port that is connnected to an end node.
- The VLAN name that the end node belongs to.
- The VLAN ID that the end node belongs to.
- A status code for the interface.

The status code indicates either `Success`, if NNMi successfully retrieved the end node information, or an error code. The display is in Comma Separated Value (CSV) format. If any of the values are not present then NNMi displays the value `-1` instead. NNMi indicates the completion of the script by displaying a CSV with all the values being `-1`.

NNMi initially displays a header naming each of the values in the CSV. There is an option to redirect the output to a file. The file can then be imported into Microsoft™ Excel.

Parameters

`nnmfindattachedswport.ovpl` supports the following parameters:

`-u <user name>`

Supply an NNMi user name to use when running the script. The user could be system, an administrator or a Level 2 operator. Required unless a nnm.properties(4) file exists.

`-p <password>`

Supply the password for the user. Required unless a nnm.properties(4) file exists.

`-i <end node file>`

Use *-i* to provide an input file containing a list of end nodes. NNMi then searches for the switch port connected to each end node. The file should have only one entry per line. The value on each line could be a MAC address, an IP address or a hostname. The MAC address needs to be specified in upper case with no leading `0x` or `0X`. This parameter is required if the *-n* is not specified.

`-n <end node>`

Use the *-n* to provide an end node for the **nnmfindattachedswport.ovpl** script. The **nnmfindattachedswport.ovpl** script finds the switch port that is connnected to this end node. The end node could be a MAC address, an IP address or a hostname. The MAC address needs to be specified in upper case with no leading `0x` or `0X`.

`-o <output file>`

Use *-o* to supply a file name to which the output of the script will be redirected.

EXAMPLES

An example of an input file is given below:

```
10.45.130.2
# this is a comment line
con5.acme.com
000087D064CB
10.12.149.4
laserj.acme.com
```

An example of the display looks similar to the following:

```
EndNode,SwitchName,IfName,VLANName,VLANId,StatusCode
10.45.130.2,-1,-1,-1,-1,UNABLE_TO_LOCATE_ENTRY_IN_FDB
con5.acme.com,-1,-1,-1,-1,UNABLE_TO_LOCATE_ENTRY_IN_ARP_CACHE
000087D064CB,10.45.130.143,2/1,Network_B_IPv4,4,SUCCESS
10.12.149.4,sw1-loop0.acme.com,Fa2/21,VLAN0490,490,SUCCESS
laserj.acme.com,sw1-loop0.acme.com,Fa2/12,mpls-intercon,169,SUCCESS
-1,-1,-1,-1,-1,-1
```

AUTHOR

`nnmfindattachedswport.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmversion.ovpl — display the version of Network Node Manager

SYNOPSIS

`nnmversion.ovpl`

DESCRIPTION

`nnmversion.ovpl` can be used to display the version of Network Node Manager that is installed. It also displays the patch number of the NNM patch that is installed. If no patches are installed, it displays a message indicating that no patches are listed.

Parameters

`nnmversion.ovpl` does not have any options.

EXAMPLES

Print out the version of NNM and any installed patches.

```
nnmversion.ovpl
```

AUTHOR

`nnmversion.ovpl` was developed by Hewlett-Packard Company.

FILES

The following files store product version and patch information:

Windows: *`data_dir`*`\NNMVersionInfo`

UNIX: `/var/opt/OV/NNMVersionInfo`

Return to Reference Pages Index

Name

ovjbosspath.ovpl — script to determine where jboss is installed

SYNOPSIS

`ovjbosspath.ovpl`

DESCRIPTION

`ovjbosspath.ovpl` is a command used by scripts to determine where jboss is used. jboss is the underlying application architecture used by `ovjboss`. Although `ovjboss` knows where jboss resides, jboss requires that certain program files (jar files) be located in underlying directories of jboss. Because other Java applications require access to these files, this provides a standard method for retrieving the base directory path.

`ovjbosspath.ovpl` is used to eliminate hard-coded paths in other applications.

Parameters

None.

EXAMPLES

On Windows with the installation in the directory `C:\Program Files(x86)\HP OpenView`, running `C:\Program Files(x86)\HP OpenView\bin\ovjbosspath.ovpl` returns the following:

```
C:/Program Files(x86)/HP OpenView/NNM
```

This enables other applications to find jar files that exist under this directory structure, such as the following:

```
C:/Program Files(x86)/HP OpenView/NNM/lib/nms-licensing-api.jar
```

AUTHOR

`ovjbosspath.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

ovjrepath.ovpl — script to determine the version of JDK to use

SYNOPSIS

`ovjrepath.ovpl`

DESCRIPTION

`ovjrepath.ovpl` is a command used by scripts to determine the version of the JDK to use. Given multiple products being installed on the system, there can be multiple JDK versions installed. These versions are not guaranteed to be compatible with Network Node Manager (NNM). This script encapsulates this problem by ensuring the correct JDK is being used.

NOTE: NNM replaces JDKs from time to time. This script enables other scripts to use the new JDK without being changed.

Parameters

None.

EXAMPLES

On Windows with the installation in the directory `C:\Program Files\HP OpenView`, running `C:\Program Files\HP OpenView\bin\ovjrepath.ovpl` returns the following:

```
C:/Program Files/HP OpenView/nonOV/jdk/b
```

This enables scripts that others are writing to use the correct JDK.

AUTHOR

`ovjrepath.ovpl` was developed by Hewlett-Packard Company.

FILES

`$InstallDir/nonOV/jdk`

Directory where JDKs are installed.

Return to Reference Pages Index

Name

ovstatus — report status of NNM managed processes

SYNOPSIS

`ovstatus` [[-c] [-d] [-v] [*managed_process_names*...]]

DESCRIPTION

`ovstatus` reports the current status of the NNM managed processes. `ovstatus` sends a status request (`OVS_REQ_STATUS`) to the process management process (UNIX operating system) or service (Windows operating system), `ovspmd`. If called with one or more *managed_process_name* arguments, it reports the status for the designated managed processes. If called with no arguments, it reports the status of all managed processes that have been added to the NNM startup file (SUF), including `ovspmd` itself.

Unlike `ovstart`, `ovstatus` does *not* start `ovspmd` if it is not already running.

The managed processes are configured by `ovaddobj` from information in Local Registration Files (see lrf(4)). A managed process is named by the first field in the LRF describing it.

Parameters

`ovstatus` recognizes the option described below. The first argument that is not an option, and any succeeding arguments, are interpreted as names of managed processes for which to report status, and are passed to `ovspmd` in the status request.

`-c`

Output one status line for each managed process.

`-d`

Report the important stages in its processing, including contacting and sending the status request to `ovspmd`, and closing the communication channel.

`-v`

Print verbose messages from managed processes. In particular, this option displays the verbose message from `ovuispmd` describing all current `ovw` sessions.

RETURN VALUE

`ovstatus` normally exits with the status `0` (zero). It returns a non-zero status only if there is a system problem, such as `ovspmd` not running.

DIAGNOSTICS

ovstatus reports certain command-line errors (in particular, too many arguments) and system errors. The messages are prefixed with ovstatus:, and are intended to be self-explanatory. ovstatus also outputs error messages received from ovspmd. These messages are prefixed with ovspmd:. ovstatus ignores unrecognized options.

ovstatus reports the known state of all OVs_WELL_BEHAVED and OVs_NON_WELL_BEHAVED processes. OVs_DAEMON processes run outside of ovspmd control. They report a PID, a state of unknown, and a final message of Does not communicate with ovspmd , as ovspmd cannot track these processes.

Note that ovspmd can process multiple requests (ovstart, ovstop, or ovstatus) at a time. If any of these commands is being handled, the new request will be queued by type until the previous command has completed.

AUTHOR

ovstatus was developed by the Hewlett-Packard Company.

FILES

The environment variables below represent universal pathnames that are established according to your shell and platform requirements. See the nnm.envvars reference page (or the UNIX manpage) for information about using environment variables for the following files:

Windows: %NNM_BIN%\ovstatus

Windows: %NNM_BIN%\ovspmd

UNIX: $NNM_BIN/ovstatus

UNIX: $NNM_BIN/ovspmd

EXTERNAL INFLUENCES

Environmental Variables

$LANG provides a default value if the internationalization variables, LC_ALL, LC_CTYPE, and LC_MESSAGES are unset, null, or invalid.

If $LANG is unset, null, or invalid, the default value of C (or English_UnitedStates.1252 on Windows) is used.

LC_ALL (or $LANG) determines the locale of all other processes started by ovspmd .

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

`LC_MESSAGES` determines the language in which messages are displayed.

SEE ALSO

ovstart(1M), ovstop(1M), ovaddobj(1M), ovdelobj(1M), ovspmd(1M), nnmcluster(1M).

Return to Reference Pages Index

Name

nnmdumpevents — dump the contents of the pre-NNM Release 8.0 event database

SYNOPSIS

`nnmdumpevents` [-f *`fileName`*] [-t] [-l *`minutes`*] [-d *`dbPath`*] [-x *`fileName`*] [-s] [-c]

DESCRIPTION

`nnmdumpevents` dumps the contents of the pre-NNM Release 8.0 event database. You can dump the entire event log, the contents of a particular stream log, or the contents of a particular correlation log. The format of the resulting file is that of the pre-NNM Release 6.0 `trapd.log` file.

The `nnmdumpevents` command dumps the contents of the event store associated with the legacy `pmd` process, and would only be used for viewing events for NNM 6.x, NNM 7.x, or both that are forwarded to the NNMi management station. The `nnmdumpevents` command does not dump traps. Use the `nnmtrapdump.ovpl` script for dumping traps. See the nnmtrapdump.ovpl(1M) reference page, or the UNIX manpage for more information.

Parameters

`nnmdumpevents` supports the following options:

`-d` *`dbPath`*

Dumps the contents of the event database contained in the directory indicated by *`dbPath`*, as opposed to dumping the default database.

`-t`

Tails the output (continuously watches for new events without returning).

`-l` *`minutes`*

Dumps the contents of the event database, starting from the last few minutes instead of from the beginning of the database. This option is useful with the `-t` option.

`-f` *`fileName`*

Writes the output of the command to the named file. If you do not specify this option, the output of the command is written to standard output. The format of the output is in the obsoleted `trapd.log` format.

`-x` *`fileName`*

Used with the `-d` option, where the `-d` option specifies the path of the directory containing the statelog file, and the `-x` option specifies the statelog file name to be used for dumping the events.

`-s streamName`

Dumps the contents of the named stream log. The format of the output is in the obsoleted `trapd.log` format.

`-c streamName`

Dumps the contents of the correlation log associated with the *`streamName`* to be dumped. The event uuids of the parent and correlated child event are dumped, as are the contents of the child event.

If you do not specify either the `-s` or `-c` option, the entire contents of the event log is dumped in the `trapd.log` format.

RETURN VALUE

If the dump of the database was successful, the `nnmdumpevents` command exits with a return value of `0` (zero). Otherwise, a non-zero value is returned.

AUTHOR

`nnmdumpevents` was developed by Hewlett-Packard Company.

FILES

The environment variables below represent universal pathnames that are established according to your shell and platform requirements. See the nnm.envvars(1) reference page for information on universal pathnames for your platform and shell.

The `nnmdumpevents` command is located in `%NNM_BIN%\nnmdumpevents` (Windows) or `$NNM_BIN/nnmdumpevents` (UNIX).

SEE ALSO

nnmtrapdump.ovpl(1)

Return to Reference Pages Index

Name

nnmsnmpnotify.ovpl — issue an SNMP notification (Trap or Inform request)

SYNOPSIS

`nnmsnmpnotify.ovpl` [-v *`version`*] [-c *`community`*] [-p *`port(default:162)`*] [-A] [-t *`timeout`*] [-r *`retries`*] [-d] [-T] [-a *`agent_addr`*] [-e *`enterprise`*] *`node trap-oid variable type value`* [*`variable type value`*]...

DESCRIPTION

If you frequently run NNMi command line tools, create an `nnm.properties` file containing your username and password. Doing so permits you to run many NNMi command line tools and scripts without entering a username and password. Place the `nnm.properties` file in a `.nnm` subdirectory within your home directory. For example, you might place the `nnm.properties` file you create in the *`drive`*`:\Documents and Settings\`*`username`*`\.nnm\` (Windows) or `~/.nnm` (UNIX) directory.

The `nnmsnmpnotify.ovpl` script sends an SNMP notification request to notify another system of an event on the local system. You can use options with the `nnmsnmpnotify.ovpl` script to acknowledge (SNMPv2 Inform) or unacknowledge (SNMPv1 or SNMPv2 Trap) the notification. You cannot send acknowledged notifications to systems that support only SNMP Version 1.

By default, the notification is unacknowledged. The `nnmsnmpnotify.ovpl` script sends an SNMP Version 1 or SNMP Version 2 Trap depending on the protocol version you specify. When you use the default version of the `nnmsnmpnotify.ovpl` script, it terminates immediately after sending the SNMP Trap request. There is no confirmation that the notification reached the destination system.

Use the `-A` option to send an acknowledged notification. The `nnmsnmpnotify.ovpl` script sends an SNMP Version 2 Inform request to the destination system. It waits for the corresponding acknowledgment, and retransmits an SNMP Version 2 Inform request if necessary. If an SNMP Version 2 Inform request retransmission occurs, the `nnmsnmpnotify.ovpl` script uses the *`timeout`* and *`retry`* values you specify on the command line. If the `nnmsnmpnotify.ovpl` script displays an acknowledgment within the time period and retry attempts you specify, you know the notification reached the destination system. If the `nnmsnmpnotify.ovpl` script does not display an acknowledgment within the time period and retry attempts you specify, the notification did not reach the destination system.

`node` can be an IP-addressable system that supports SNMP. You can identify IP nodes by Internet address or hostname. You can supply *`node`* in Internet address form or hostname form. If you supply an empty string ("") to the `nnmsnmpnotify.ovpl` script instead of a node, the script uses localhost as the destination.

Specify the trap type as an object identifier in the *`trap_oid`* command-line argument. You must identify all notifications using the object identifier form. You can supply notifications defined in the SNMPv2 MIB or in a vendor-specific SNMPv1 MIB directly to the `nnmsnmpnotify.ovpl` script. However, you must convert traps defined in a vendor-specific SNMPv1 MIB to the object identifier form before supplying them to the `nnmsnmpnotify.ovpl` script. For an SNMP Version 1 trap, if you supply an empty string ("") instead of a *`trap_oid`*, the Generic trap type value is set to 6 and the Specific trap type value is set to 0. For an SNMP Version 2 trap, if you supply an empty string ("") instead of a *`trap_oid`* the *`trap_oid`* variable binding is

not set.

When providing trap object identifiers to the `nnmsnmpnotify.ovpl` script, follow these guidelines:

- Use the corresponding object identifiers defined in RFC 1907 to generate a trap for any of the six generic SNMP traps: `coldStart`, `warmStart`, `linkDown`, `linkUp`, `authenticationFailure`, and `egpNeighborLoss`. For example, use the `1.3.6.1.6.3.1.1.5.1` trap OID to generate a `coldStart` trap. .
- To generate a trap that is not SNMP-generic but is defined in SNMPv2 form, use the `NOTIFICATION-TYPE` identifier from the SNMPv2-compatible MIB.
- To generate a trap that is not SNMP-generic but is defined in SNMPv1 form, determine the trap enterprise and specific numbers from the SNMPv1-compatible MIB. From these trap enterprise and specific numbers, construct an object identifier in the *`enterprise.0.specific field`* form. For example, consider a vendor-specific MIB for a device test. The MIB defines a trap with enterprise `1.3.6.1.4.1.11.2.17.1` and specific trap field `4`. The resulting trap object identifier would be `1.3.6.1.4.1.11.2.17.1.0.4`.

The `nnmsnmpnotify.ovpl` script passes data to the remote node as a triple of *`variable,type,value`*. Supply one or more triples to the `nnmsnmpnotify.ovpl` script as command-line arguments.

Each variable is an object instance identifier in either dotted decimal format or mnemonic string format. For example, you can use either the `.1.3.6.1.4.1.11.2.17.2.1.0` or the `openViewSourceId.0` format.

Each *`type`* must be one of the following types:

INTEGER

INTEGER32

IPADDRESS

COUNTER

COUNTER32

COUNTER64 (for SNMPv2c or v3 capable remote nodes)

GAUGE

GAUGE32

OBJECTIDENTIFIER

OCTETSTRING

OCTETSTRINGASCII

OCTETSTRINGHEX

OCTETSTRINGOCTAL

OPAQUE

OPAQUEASCII

OPAQUEHEX

OPAQUEOCTAL

TIMETICKS

UNSIGNED32

For a complete description of each *type*, refer to *RFC 1155* and *RFC 1902*.

The *value* parameter must be valid for the type specified. When using a type that requires a hexadecimal or octal value, you must fully define each byte of the value. For example, if you specify `fff` (or `17377`), it is missing a byte, and will not work. Use `0fff` (or `017377`) instead. You must specify a *value* on the command line. *value* must not be larger than 512 bytes.

Parameters

`-v` *version*

Requests the `nnmsnmpnotify.ovpl` script to use a specific version of SNMP to communicate with the remote node. Valid choices for *version* are `1`, `2c`, or `3`.

`-c` *community*

Specifies the community string to use for authentication on the remote node.

Note: If the community string contains characters the shell interferes with, use one or more escape symbols or quotation marks as required.

`-p` *port*

Specifies the port to use to communicate with the remote node.

`-t` *timeout*

Specifies a timeout period, in tenths of seconds, to wait for an acknowledgment when using an SNMP Version 2 Inform request. This option is only valid when used with the `-A` option.

`-r` *retries*

Specifies the number of retransmissions to attempt when no acknowledgment is received when using an SNMP Version 2 Inform request. This option is only valid when used with the `-A` option.

`-d`

Dump ASN.1 packet trace

`-T`

Prints the OID in dotted decimal format.

`-a` *agent_add*

Overrides the local host as the source of the notification with the given agent address. *agent_addr* must be an IP address or hostname.

`-e` *enterprise*

Overrides the default enterprise object identifier for the notification with the given *enterprise* value.

EXAMPLES

The following command sends an SNMP link down Inform request to the node `v2c_node`:

```
nnmsnmpnotify.ovpl -A -v2c v2c_node .1.3.6.1.6.3.1.1.5.3
```

The following command sends an SNMP link down trap request to the node `v1_node` with the agent address set to `agent`:

```
nnmsnmpnotify.ovpl -a agent v1_node .1.3.6.1.6.3.1.1.5.3
```

AUTHOR

`nnmsnmpnotify.ovpl` was developed by Hewlett-Packard Company.

FILES

The environment variable below represents a universal path that is established according to your shell and platform requirements:

Windows: `%NNM_BIN%\nnmsnmpnotify.ovpl`

UNIX: `$NNM_BIN/nnmsnmpnotify.ovpl`

SEE ALSO

nnmsnmpwalk.ovpl(1M), nnmsnmpset.ovpl(1M), nnmsnmpbulk.ovpl(1M).

RFC 1155, 1157, 1212: SNMP Version 1.

RFC 1901 - 1908, 2576, 2578, 3416 - 3418: SNMP Version 2.

RFC 3411 - 3415: SNMP Version 3.

EXTERNAL INFLUENCES

Environmental Variables

`$LANG` determines the language in which messages appear. If `$LANG` is not specified or is set to an empty

string, a default of `C` is used instead of `$LANG`. If any internationalization variable contains an invalid setting, `nnmsnmpnotify.ovpl` behaves as if all internationalization variables are set to `C`.

International Code Set Support

Supports single-byte and multiple-byte character code sets.

NOTE: SNMP MIB values of the type `octetstringascii` are restricted to NVT-ASCII.

Return to Reference Pages Index

Name

nmsdbmgr — controls the NNMi embedded database, including periodic database connectivity testing

SYNOPSIS

`nmsdbmgr` [-ovspmd] [-start] [-test] [-initnmsdb] [-stop] [-status] [-kill]

DESCRIPTION

`nmsdbmgr` is a program that provides an interface for the `ovspmd` process to control the NNMi embedded database. The `nmsdbmgr` program enables the `ovspmd` process to start, stop, and check the status of the embedded database.

While running, the program tests database connectivity every five minutes and updates the status message reported to the `ovspmd` process, depending on the results of the connectivity test. If the database test succeeds, the message reported is `Database Available`. If the connectivity test fails, the message reported is `Data Warehouse Inaccessible`. If you see the failure message, you may experience database-related problems in NNMi, such as the failure to retrieve and display the node inventory in the NNMi console.

While it is technically possible to run this command independently of the `ovspmd` process, doing so is highly discouraged and could cause unexpected results.

NOTE: If you examine the process table while the `nmsdbmgr` program is running, notice that this program spawns many child processes named `postgres` or `postgres.exe`, depending on the platform. These child processes represent the embedded database itself. Having many instances of them is normal.

EXAMPLES

Normal usage of this command is indirect, using the `ovspmd` process.

Typical usage starts or stops the process as follows:

```
# ovstart –c nmsdbmgr
```

```
# ovstop –c nmsdbmgr
```

```
# ovstatus –c nmsdbmgr
```

AUTHOR

`nmsdbmgr` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_DB%\Postgres`

Windows: `%NNMInstallDir%\nonOV\Postgres`

UNIX: `$NNM_DB/Postgres`

UNIX: `$NNMInstallDir/nonOV/Postgres`

SEE ALSO

ovspmd(1M), ovstart(1M), ovstop(1M), ovstatus(1M).

Return to Reference Pages Index

Name

nnmaction — wrapper process for the HP NNMi Action Server (action server).

SYNOPSIS

`nnmaction`

DESCRIPTION

`nnmaction` is a process referred to as the action server. It is managed by the `ovspmd` process. You can pass arguments to the action server by adding entries to the `nnmaction.properties` file.

Never run the `nnmaction` command manually. The `ovspmd` process starts and manages the `nnmaction` process. To restart the `nnmaction` process, run the `ovstop nnmaction` command followed by the `ovstart nnmaction` command. Run the `ovstatus nnmaction` command to determine the status of the `nnmaction` process.

You must be logged on as root or ar administrator to run the `ovstart` or `ovstop` commands.

AUTHOR

`nnmaction` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_SHARED_CONF%\props\nnmaction.properties`

Parameter file used by the action server.

UNIX: `$NNM_SHARED_CONF/props/nnmaction.properties`

Parameter file used by the action server.

SEE ALSO

ovspmd(1M), ovstart(1M), ovstop(1M), ovstatus(1M)

Return to Reference Pages Index

Name

nnmbackup.ovpl — script used to back up NNMi data and files

SYNOPSIS

`nnmbackup.ovpl` [-?|-h|-help] [-type (online|offline)] [-scope (config|topology|events|all)] [-force] [-archive] [-noTimeStamp] -target <*`directory`*>

DESCRIPTION

`nnmbackup.ovpl` is the main backup script for NNMi. For NNMi installations using an embedded database, the `nnmbackup.ovpl` script determines the directories and tables to back up using the `%NNM_DATA%\shared\nnm\backup.properties` (Windows) or `$NNM_DATA/shared/nnm/backup.properties` (UNIX) file. The `backup.properties` file also defines any files and directories that require extra processing during the restore phase. The `nnmbackup.ovpl` script accepts arguments to determine things like backup scope (`config`, `topology`, `events`, `all`), backup location, and backup type (`online` or `offline`).

If you plan to use the `nnmbackup.ovpl` script to create an NNMi backup, then use the `nnmrestore.ovpl` script to place database records on a second NNMi management server, both NNMi management servers must have the same type of operating system and NNMi version and patch level. Placing the backup data from one NNMi management server onto a second NNMi management server means that both servers have the same database UUID. After you restore NNMi on the second NNMi management server, uninstall NNMi from the original NNMi management server.

Before running the `nnmbackup.ovpl` script, make sure you have adequate storage space in the target directory. For most NNMi installations, if you have enough space to store the contents of the NNMi installation, `%NNM_DATA%` (Windows), or `$NNM_DATA` (UNIX) directories, you should have adequate storage space. Check the available storage space in the following locations:

- *Windows*: `%NnmInstallDir%`
- *UNIX*:`/opt/OV` (UNIX)

If you selected the embedded database option during the NNMi installation, you can find the embedded database data storage in the `%NNM_DATA%\shared\nnm\databases\Postgres` (Windows) or `$NNM_DATA/shared/nnm/databases/Postgres` (UNIX) directory.

The target directory contains all of the files applicable for the backup options you have selected, or a single tar file if you use the `–archive` option. Each backup operation stores files in a parent directory called `nnm-bak-<`*`TIMESTAMP`*`>` inside of target directory. Any database operations occurring during the backup are included in the backup. You can compress the files after the backup completes.

Files that require extra processing during the restore phase are stored with their full paths beneath the `target_directory/special_files/handling_routine` directory. During the restore phase, NNMi selects files for exclusion, restoration, or merge. For more information, see the `nnmrestore.ovpl` reference page, or the UNIX manpage.

The `nnmbackup.ovpl` script includes the necessary data to perform a restore operation. You must be logged

in as administrator on Windows NNMi management servers or root on UNIX NNMi management servers to run the `nnmbackup.ovpl` script.

Caution

Database backups performed by the `nnmbackup.ovpl` script only apply to the embedded database. If you chose a different database at install time, the table data is not backed up using this script. File-system backups work regardless of the database type. For details about how to back up NNMi data if you select a different database at install time, see the *NNMi Deployment Reference* (available at http://h20230.www2.hp.com/selfsolve/manuals).

Parameters

The `nnmbackup.ovpl` script supports the following options:

`-type (online|offline)`

This option determines the type of backup to be performed. If you specify the `online` option, both NNMi and the `nmsdbmgr` process must be running before running the `nnmbackup.ovpl` script. If you specify the `offline` option, completely stop NNMi before running the `nnmbackup.ovpl` script.

`-scope (config|topology|events|all)`

This option determines the scope of the backup operation. There are two types of data that the `nnmbackup.ovpl` script backs up: files in the file system and tables in the database. The `-scope` option value for files in the file system is always applicable, regardless of the backup type you choose (see the `–type` option). However, the `-scope` option value for tables in the database is applicable only when you run an online backup using the `-type online`) option. For offline scoped backups, you get the entire contents of the database, not just the scope you request. For this reason, HP recommends that you do not define `scope` when doing offline backups (the default is `all`). The scopes available are `config`, `topology`, `events`, and `all`. Each scope includes all of the data and files from the previous scope (`all` → `events` → `topology` → `config`). The `%NNM_DATA%\shared\nnm\backup.properties` (Windows) or `$NNM_DATA/shared/nnm/backup.properties` (UNIX) file contains a list of the files and tables backed up for each scope.

`-force`

If you use the `-force` option, the `nnmbackup.ovpl` script starts and stops NNMi based on the type of backup you requested. For online backups, if NNMi is not running, the `nnmbackup.ovpl` script starts the `nmsdbmgr` process (required for backups). For offline backups, if NNMi is running, the command stops all NNMi processes.

`-archive`

If you provide the `-archive` option, the `nnmbackup.ovpl` script stores the backup files in a tar file in the target directory.

`-noTimeStamp`

With this option, the `nnmbackup.ovpl` script stores the backup files in a target directory without a timestamp in the name, i.e. just "nnm-bak" or "nnm-bak.tar". Any previous backup which exists with that same folder/file name will be renamed to have ".previous" suffix. If there is already a ".previous" backup, it will be deleted. This option is provided to allow for daily backups, keeping latest two successful backups, and not keeping every backup that has been performed, over time, to reduce disk

usage.

`-target <directory>`

Specifies the output directory where you want the backup files stored. The `nnmbackup.ovpl` script creates a parent directory named `nnm-bak-<TIMESTAMP>` inside the target directory where all backup files are stored. If the `archive` option is present, the `nnmbackup.ovpl` script creates a temporary directory, `nnm-bak-<TIMESTAMP>`, then replaces this directory with a tar file using the same name as the temporary directory.

`-?|-h|-help`

Display command usage.

EXAMPLES

Suppose you want to save the NNMi configuration before discovering your network, but do not want to save the discovery results. To do this, you might run a backup using the `online` and `config` options:

```
#./nnmbackup.ovpl –type online –scope config –target /tmp/bak/config
```

Suppose you want to save the NNMi configuration, discovered topology, and event data without bringing the application down. To do this, you might run a backup using the `online`, `events`, and `force` options:

```
#./nnmbackup.ovpl –type online –scope events –target /tmp/bak/evt
```

Suppose you want to run a regularly scheduled backup. To do this you might perform a backup using the `offline` and `full` options:

```
#./nnmbackup.ovpl –force –type offline –scope all –target /tmp/bak/all
```

AUTHOR

The `nnmbackup.ovpl` script was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_DATA%\shared\nnm\backup.properties`

UNIX: `$NNM_DATA/shared/nnm/backup.properties`

SEE ALSO

nnmrestore.ovpl(1).

Return to Reference Pages Index

Name

nnmbackupembdb.ovpl — make a full backup of the NNMi embedded database.

SYNOPSIS

```
nnmbackupembdb.ovpl [-?|-h|-help] -target <target directory> [-force]
```

DESCRIPTION

Use the `nnmbackupembdb.ovpl` script to make a full backup of the NNMi embedded database. The contents of the backup are uncompressed and stored in a file you specify. This file must be used to restore the NNMi embedded database by using the `nnmrestoreembdb.ovpl` script.

If you plan to use the `nnmbackupembdb.ovpl` script to create a backup of the NNMi embedded database, then use the `nnmrestoreembdb.ovpl` script to place embedded database records on a second NNMi management server, both NNMi management servers must have the same type of operating system and NNMi version and patch level.

Before running the `nnmbackupembdb.ovpl` script, make sure you have adequate storage space in the target directory. Look for the embedded database data storage in the `%NNM_DATA%\shared\nnm\databases\Postgres` (Windows) or `$NNM_DATA/shared/nnm/databases/Postgres` (UNIX) directory. Check the size of this directory to make sure the target directory has sufficient space to store the backup file. Compress the file after backup if necessary.

The backup file is created in the target directory and contains all data stored in the database at the start of the backup operation. Any statements run in the database while the backup is running are not included in the backup.

You can run this script while NNMi is running; however, you may experience temporary performance degradation. At a minimum, the `nmsdbmgr` process must be running for the script to be successful. If you use the `-force` option, the script starts the `nmsdbmgr` process (if it is not running), and suppresses any interactive messages.

You must be logged in as administrator on Windows systems or root on UNIX systems to run this script.

Parameters

`-target <directory>`

Directory name in which to store the backup file. (Will be created if it does not exist)

`-force`

If you provide this option, the script starts the `nmsdbmgr` process if it is not currently running.

`-?|-h|-help`

Display command usage.

EXAMPLES

You can use this script to run an *on-demand* backup, or you can include running the script as a task in regularly scheduled backup scripts. Run the script as follows:

```
# nnmbackupembdb.ovpl -target /backups/nnm
```

You'll see these messages:

```
WARNING: Running this command while NNM is running, while allowed, will cause
         temporary performance problems. At a minimum, please make sure the
         nmsdbmgr process is running (ovstart nmsdbmgr).

Are you sure you want to run a full database backup now? [n] y

Ok, performing full embedded database backup...

NNM embedded database successfully backed up to /backups/nnm/nnm-bak.2009092906.pgd.
#
```

AUTHOR

`nnmbackupembdb.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_BIN%\nnmbackupembdb.ovpl`

UNIX: `$NNM_BIN/nnmbackupembdb.ovpl`

SEE ALSO

ovstart(1M), ovstop(1M), ovstatus(1M), nmsdbmgr(1M), nnmrestoreembdb.ovpl(1M).

Return to Reference Pages Index

Name

nnmcertmerge.ovpl — Automation of trust and key store merges into NNMi certificate stores

SYNOPSIS

`nnmcertmerge.ovpl` [-?|-h|-help] [-keystore <*`file`*> -truststore <*`file`*>]|[-directory <*`directory`*>]

DESCRIPTION

Use the `nnmcertmerge.ovpl` script to automate certificate store merges into the `nnm.keystore` and `nnm.truststore` files. The `nnmcertmerge.ovpl` script simplifies the task of merging all certificates when NNMi is using the Global Network Management, High Availability, or application failover features.

You must be logged in as `administrator` on Windows systems or `root` on UNIX systems or as to run this script.

Parameters

`nnmcertmerge.ovpl` supports the following options:

`-keystore` <*`file`*>

If you provide this option, the target file is merged into the `nnm.keystore` file. This option can be provide at the same time as the `-truststore` option.

`-truststore` <*`file`*>

If you provide this option, the target file is merged into the `nnm.truststore` file. This option can be provide at the same time as the `-keystore` option.

`-directory` <*`directory`*>

You must use the `-directory` option by itself. If you provide this option, the script handles all files in the target directory as follows:

The script merges all files ending in `.keystore` into the `nnm.keystore` file.

The script merges all files ending in `.truststore` into the `nnm.truststore` file.

`-?|-h|-help`

Displays the command usage.

EXAMPLES

Merge a keystore with NNMi:

```
nnmcertmerge.ovpl -keystore /tmp/hostA.keystore
```

Merge a truststore with NNMi:

```
nnmcertmerge.ovpl -truststore /tmp/hostA.truststore
```

Merge a keystore and truststore with NNMi:

```
nnmcertmerge.ovpl -keystore /tmp/hostA.keystore -truststore /tmp/hostA.truststore
```

Merge a set of keystores and truststores with NNMi:

```
nnmcertmerge.ovpl -directory /tmp/AppFailoverHosts/
```

AUTHOR

`nnmcertmerge.ovpl` was developed by Hewlett-Packard Company.

FILES

The `nnmcertmerge.ovpl` script resides in the `%NNM_BIN%` directory (Windows) or the `$NNM_BIN` directory (UNIX).

Return to Reference Pages Index

Name

nnmchangedbpw.ovpl — change the user name and password used to authenticate with the NNMi database.

SYNOPSIS

`nnmchangedbpw.ovpl`

DESCRIPTION

Use the `nnmchangedbpw.ovpl` script to change the user name and password NNMi uses to connect to the database. This script is useful when database passwords expire, or if you need to change the database user name. Before running this script, run the **`ovstop`** command to stop NNMi.

> **Note**
>
> To avoid adverse behavior after changing the database user name and password, you must stop NNMi before running the `nnmchangedbpw.ovpl` script. You can run this script while NNMi is running, but any new database connections created by NNMi fail during authentication.

You must be logged in as `administrator` on Windows systems or `root` on UNIX systems to run this script.

Parameters

No supported parameters.

EXAMPLES

Use the `nnmchangedbpw.ovpl` script if your organization requires frequent password changes for the NNMi database, or to change the user name NNMi uses to connect to the NNMi database.

The `nnmchangedbpw.ovpl` script does not display the password as you type it in.

When using the `nnmchangedbpw.ovpl` script, you should see the following messages:

```
# nnmchangedbpw.ovpl

WARNING: Please make sure NNM has been stopped before running this tool.
         Failure to stop NNM could result in unexpected database failures.

Have you stopped NNM (ovstop)? [n] y
Thank you!

Please provide a database user account name.
user: mydbuser

Please provide the password for database user account.
password: mynewpw
enter password again: mynewpw
```

```
User/Password changed successfully!
#
```

AUTHOR

`nnmchangedbpw.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_BIN%\nnmchangedbpw.ovpl`

UNIX: `$NNM_BIN/nnmchangedbpw.ovpl`

SEE ALSO

ovstart(1M), ovstop(1M), ovstatus(1M).

Return to Reference Pages Index

Name

nnmchangeembdbpw.ovpl — change the password used to authenticate with the NNMi embedded database.

SYNOPSIS

`nnmchangeembdbpw.ovpl`

DESCRIPTION

Use the `nnmchangeembdbpw.ovpl` script to change the password NNMi uses to connect to the embedded database. This script is useful if you installed NNMi with the embedded database option. Only use this script if you want to change the default password that was created for the embedded database during installation. If you do not run this script, NNMi functions normally.

Before running the `nnmchangeembdbpw.ovpl` script, do the following:

- Run the **`ovstop`** command to stop NNMi.
- Run the **`ovstart nmsdbmgr`** command to start the `nmsdbmgr` process.

  Note

  If the `nmsdbmgr` process is not running, the `nnmchangeembdbpw.ovpl` script fails.

After the `nnmchangeembdbpw.ovpl` script finishes, the embedded database password is changed to the value supplied through the interactive prompts, and the NNMi management server is reconfigured to use the new password to connect to the database.

You must be logged in as `administrator` on Windows systems or `root` on UNIX systems to run the `nnmchangeembdbpw.ovpl` script.

Parameters

No supported parameters.

EXAMPLES

Use the `nnmchangeembdbpw.ovpl` script if your organization requires frequent password changes for the NNMi embedded database, or to change the user name NNMi uses to connect to the embedded database.

The `nnmchangeembdbpw.ovpl` script does not display the password as you type it in.

When using the `nnmchangeembdbpw.ovpl` script, you should see the following messages:

```
# nnmchangeembdbpw.ovpl
```

```
WARNING: Stop NNMi and all NNMi processes before running this tool.
         Failure to stop NNMi could result in unexpected database failures.
         After stopping NNMi, run ovstart nmsdbmgr to start only the database.

Have you stopped NNM (ovstop)? [n] y
Thank you!

Provide a new password for the embedded database user account.
password: mynewpw
enter password again: mynewpw

Password changed successfully!
#
```

AUTHOR

`nnmchangeembdbpw.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_BIN%\nnmchangeembdbpw.ovpl`

UNIX: `$NNM_BIN/nnmchangeembdbpw.ovpl`

SEE ALSO

ovstart(1M), ovstop(1M), ovstatus(1M), nmsdbmgr(1M).

Return to Reference Pages Index

Name

nnmchangesyspw.ovpl — change the password for the system account normally set during NNMi installation.

SYNOPSIS

`nnmchangesyspw.ovpl`

DESCRIPTION

You can use the `nnmchangesyspw.ovpl` script to change the NNMi system password. The NNMi system password is normally set during installation and used for recovery purposes. Only use this command if you want to reset the system password that was set during NNMi installation.

Before running the `nnmchangesyspw.ovpl` script, run the `ovstop` command to stop NNMi. After running the `nnmchangesyspw.ovpl` script, run the `ovstart` command to start NNMi. This will ensure that the new value for the password is valid immediately.

You must be logged in as `administrator` on Windows systems or `root` on UNIX systems to run this script.

Parameters

No supported parameters.

EXAMPLES

You might want to change the system password if you have deleted all other user accounts with administrator privileges, and do not remember the system password value set during NNMi installation.

The `nnmchangesyspw.ovpl` script does not display the password as you type it in.

When using the `nnmchangesyspw.ovpl` script, you should see the following messages:

```
# nnmchangesyspw.ovpl

WARNING: This change may not take affect immediately unless NNM is
         restarted.  Please run ovstop before executing this script,
         and ovstart after execution to ensure the change is immediate.
Would you like to continue? [n] y
Thank you!

Please provide a new password for the system user account.

password: mynewpw
enter password again: mynewpw

System password changed successfully

#
```

AUTHOR

`nnmchangesyspw.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_BIN%\nnmchangesyspw.ovpl`

UNIX: `$NNM_BIN/nnmchangesyspw.ovpl`

SEE ALSO

ovstart(1M), ovstop(1M), ovstatus(1M).

Return to Reference Pages Index

Name

nnmcommconf.ovpl — display communication configuration information

SYNOPSIS

`nnmcommconf.ovpl` [-u *username*] [-p *password*] [-jndiHost *host name*] [-jndiPort *port* Default is 1099] -proto <*icmp* | *snmp*> -host <*hostname*>

DESCRIPTION

`nnmcommconf.ovpl` is a script that reads information from NNMi about how NNMi tries to communicate with a given host using a specific protocol, then displays this information. The `nnmcommconf.ovpl` script displays information based on either SNMP or ICMP protocols.

Parameters

`nnmcommconf.ovpl` recognizes the following options.

`-proto <protocol>`

protocol: SNMP or ICMP

`-host`

The name of the host you plan to retrieve information from.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost`

The server JNDI host; the default is `localhost`.

`-jndiPort`

The server JNDI port; the default is 1099.

EXAMPLES

```
nnmcommconf.ovpl -username foo -password bar -proto icmp -host baz
```

By running the command as shown above, you might get the information displayed below.

```
      address     =10.2.1.2

      timeout     =2000

      address     =10.2.1.2

      retries     =1

      enabled     =true

      region name     =default
```

```
nnmcommconf.ovpl -username foo -password bar -proto snmp -host baz
```

By running the command as shown above, you might get the information displayed below.

```
      name     =baz

      address     =10.2.1.1

      addressForced     =false

      getCommunity     =public

      timeout     =5000

      retries     =1

      port     =161

      enabled     =true

      region name     =default
```

AUTHOR

`nnmcommconf.ovpl` was developed by Hewlett-Packard Company.

FILES

`nnmcommconf.ovpl` resides in the following directory:

Windows: `%NNM_BIN%\nnmcommconf.ovpl`

UNIX: `$NNM_BIN/nnmcommconf.ovpl`

SEE ALSO

nnm.properties(4), nnmcommload.ovpl(1M).

Return to Reference Pages Index

Name

nnmcommload.ovpl — load communication settings from a comma separated file

SYNOPSIS

`nnmcommload.ovpl` [-u *`username`*] [-p *`password`*] [-jndiHost *`hostname`*] [-jndiPort *`port`* Default is 1099] -file <*`filepath`* | *`filename`*>

DESCRIPTION

Use the `nnmcommload.ovpl` script to import communication settings for a group of devices in bulk. This is useful if the community strings are managed by a change control mechanism. You can bulk insert the assignments into NNMi. Depending upon the format of the data entered in the configuration file, each assignment will show up in the NNMi console as either an individual entry in the `Regions` or `Specific Node Settings` tab of the `Communication Configuration` dialog.

If you specify hostnames as IP addresses, the `nnmcommload.ovpl` script does not resolve the IP addresses to fully qualified names. If you specify actual hostnames, the `nnmcommload.ovpl` script resolves the hostnames to their fully qualified names using DNS. This can take some time to work through large import files. For files containing more than 500 lines, the `nnmcommload.ovpl` script saves the entries to the database in batches of 500. After the `nnmcommload.ovpl` script reads 500 lines from the import file, the SNMP configuration entries for those lines will be resolved based on any existing SNMP region or default settings and saved to the database.

To perform the import, create a text file that includes the information shown below. Create one line for each device. Within each line, add the information in the order shown in the list below. Separate each value with a comma. Enter comments in lines that start with a number (`#`) character. The `nnmcommload.ovpl` script interprets the data strictly by position within the line, so you must specify a comma as a placeholder for non-specified values. You can embed commas by quoting the string. For example: "comm,string"

- Target node name or IP address (required for Specific Node Setting configuration)

  One or more Hostname Filters separated by ";;" (optional for Region configuration)

- Single Read community string (optional for Specifific Node Setting configuration)

  One or more read community strings separated by ";;" (optional for Region configuraion)

  If community string priority is desired, append the string "#PRI#" followed by the priority to each region community string. For example "public#PRI#5" would assign a priority of 5 to community string "public"

- Management address (optional for Specific Node Setting configuration)

  One or more addreass ranges separated by ";;" (optional for Region configuration)

- Write community string (optional)
- Timeout in milliseconds (optional)
- Number of retries (optional)
- Port (optional)

- Proxy address (optional)
- Proxy port (optional)
- User name (SNMP V3 optional for Specific Node Setting Configuration)

  One or more User Names separated by ";;" (SNMP V3 optional for Region Configuration)
- Context name (SNMP V3 optional for Specific Node Setting Configuration)

  One or more Context Names separated by ";;" (SNMP V3 optional for Region Configuration)
- Authentication protocol (SNMP V3 optional for Specific Node Setting Configuration - MD5|SHA)

  One or more Authentication Protocols separated by ";;" (SNMP V3 optional for Region Configuration)
- Authentication password (SNMP V3 optional for Specific Node Setting Configuration)

  One or more Authentication Passwords separated by ";;" (SNMP V3 optional for Region Configuration)
- Privacy protocol (SNMP V3 optional for Specific Node Setting Configuration - DES|3DES|AES|AES192|AES256)

  One or more Privacy Protocols separated by ";;" (SNMP V3 optional for Region Configuration)
- Privacy password (SNMP V3 optional for Specific Node Setting Configuration)

  One or more Privacy Passwords separated by ";;" (SNMP V3 optional for Region Configuration)
- Preferred SNMP version (optional - 1|2|3 for Specific Node Setting Configuration only)
- Enable SNMP Communication flag (optional - true|false)
- Enable SNMP Address Rediscovery flag (optional - true|false)
- Enable SNMP GetBulk flag (optional - true|false)
- Description (optional for either Specific Node Setting or Region Configuration)
- Enable ICMP Communication flag (optional - true|false)
- ICMP Timeout in milliseconds (optional)
- ICMP Number of retries (optional)
- Device Credential User Name(optional for Specific Node Setting Configuration)

  One or more Device Credential User Names separated by ";;" (optional for Region Configuration)
- Device Credential Password (optional for Specific Node Setting Configuration)

  One or more Device Credential Passwords separated by ";;" (optional for Region Configuration)

- Device Credential Type (optional for Specific Node Setting Configuration)

  One or more Device Credential Types separated by ";;" (optional for Region Configuration) Currently the only type supported is "Shell". There can only be one conifiguration per type per region

- Region Name (Optional for Region Configuration - Defaulted to "Region" + Ordering, e.g. Region15)
- Region Ordering (Optional for Region Configuration - Defaulted to 1 greater than the current max ordering value for existing regions)
- Minimum SNMP Security Level (Optional for Region Configuration - V1-ONLY|V1V2-ONLY|COMMUNITY|NOAUTH-NOPRIV|AUTH-NOPRIV|AUTH-PRIV) If not specified, defaulted to value of COMMUNITY

For example, all of the following entries are considered valid entries for loading a Specific Node Configuration:

```
hostname
```

```
hostname,
```

```
hostname,,
```

```
hostname,public
```

```
hostname,,10.2.2.3,,1000,2,161
```

```
node1,community,10.3.7.96,writecommunity
```

`node2, community,10.3.7.95` (the space character before community will be removed)

```
10.2.23.34,community,10.2.23.8
```

```
10.2.23.34,community,10.2.23.88,writecommunity,2000,2,161,10.56.22.199,162
```

Here's an example for loading a Region Configuration with multiple hostname filters, address ranges, read community strings, SNMPv3 configurations, and device credentials. Note also the community string priority being assigned.

```
testv3*;;cisco*.fc.usa.hp.com, public;;ntcpublic#PRI#1, 3.3.3.3;;4.4.4.4, ntcwrite,
3333,3,161,7.7.7.7,777,v3User1;;v3User2,v3Context1;;v3Context2,MD5;;SHA,authPass1;;authPass2,AES;;DES,privPass1;;privPass2,3,true,T,T,my region
description,T,3,3,user1,password1,shell,myregion,10,Community
```

Here's another example for loading a Region Configuration with multiple SNMPv3 configurations. Only the V3 and region parameters are specified.

- v3User1 is a No Authenticaton/No Privacy user
- v3User2 has MD5 Authentication and No Privacy
- v3User3 has SHA Authentication and 3DES Privacy

Note how the ";;" separators are used to skip specifying authentication and privacy information where appropriate

```
,,,,,,,,v3User1;;v3User2;;v3User3,v3Context1;;;;v3Context3,;;MD5;;SHA,;;authPass2;;authPass3,;;;;3DES,;;;;privPass3,3,,,,,,,,,,myregion,10,Community
```

Parameters

`nnmcommload.ovpl` recognizes the following options.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost` (optional)

The hostname of the server running the jboss application server. If you do not specify a hostname, the `nnmcommload.ovpl` script uses `localhost` as the default value.

`-jndiPort` (optional)

The jboss application server port. If you do not specify this port, the `nnmcommload.ovpl` script uses 1099 as the default value.

`-file` (required)

A file name or full path to a file created with the format described above containing the data to be loaded.

EXAMPLES

```
nnmcommload.ovpl -user joe -password secret import.txt

nnmcommload.ovpl -user joe -password secret C:\temp\import.txt

nnmcommload.ovpl -user joe -password secret /tmp/import.txt

nnmcommload.ovpl -user joe -password secret -jndiHost myserver -jndiPort 1117 secret import.txt
```

AUTHOR

`nnmcommload.ovpl` was developed by Hewlett-Packard Company.

FILES

`nnmcommload.ovpl` resides in the following location:

Windows: *`install_dir`*`\bin\nnmcommload.ovpl`

UNIX: `/opt/bin/nnmcommload.ovpl`

SEE ALSO

nnm.properties(4), nnmcommconf.ovpl(1M).

Return to Reference Pages Index

Name

nnmconfigexport.ovpl — Export the configuration to one or more files that can be imported on another system.

SYNOPSIS

`nnmconfigexport.ovpl` -? | -c <*`configuration`*>*`[,configuration...]`* [-a <*`author_key`*>] [-u <*`username`*> -p <*`password`*>] [-x <*`file_prefix`*>] [-f <*`output file or directory`*>]

DESCRIPTION

`nnmconfigexport.ovpl` is a Perl script that enables you to export the customized configuration to stdout, or to save it in a file.

Parameters

`nnmconfigexport.ovpl` supports the following options:

`-?`

Displays the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-c <configuration>[,configuration...]`

Exports the XML schema for the specified configuration. Use a comma separated list for multiple configurations. If there are multiple configurations, you must specify a directory when using the `-f` option.

Valid configurations:

`account`

Exports user accounts, user roles, user principals, user groups and user account mappings.

`author`

Exports authors. Can be optionally filtered using the `-a` argument.

`customCorrelation`

Exports custom correlation configuration. Can be optionally filtered using the `-a` argument.

`comm`

Exports communication configuration. Does not export SNMPv3 communication configuration and device credentials. The encryption algorithms used with this data rely on an internal key specific to the NNMi installation. It is not possible to import this data, so the data is excluded from the export.

`custpoll`

Exports custom poller configuration.

`device`

Exports device profiles. Can be optionally filtered using the `-a` argument.

`disco`

Exports discovery configuration (does not include seeds).

`discoseed`

Exports discovery seeds.

`icons`

Exports icons.

`ifgroup`

Exports interface groups.

`iftype`

Exports interface types (IfTypes). Can be optionally filtered using the `-a` argument.

`incident`

Exports incident configuration. Can be optionally filtered using the `-a` argument.

`menu`

Exports menus. Can be optionally filtered using the `-a` argument.

`menuitem`

Exports all menu items configured for the `Actions` menu. If you supply the `-a` argument, the output contains the related parent menus and sub-menus.

`mibexpr`

Exports MIB expressions. Can be optionally filtered using the `-a` argument.

`mibtypes`

Exports MIB OID Types.

`monitoring`

Exports monitoring configuration.

`nodegroup`

Exports node groups.

`ngmap`

Exports node group maps. The node host names must match between both NNMi management servers for the node coordinates to import successfully.

`oam`

Exports Overlapping Address Mappings.

`rams`

Exports one or more route analytics management server configurations.

NOTE: The `Query Password` field of a RAMS configuration will only be valid when you import it into the same NNMi installation on the same system. If you import the `Query Password` field into a different system, you will have to re-enter the `Query Password`.

`security`

Exports security groups and tenants.

`securitymappings`

Exports security group mappings.

`station`

Exports NNM 6.x/7.x management stations.

`status`

Exports the node group status configuration.

`trap`

Exports the trap logging configuration. Can be optionally filtered using the `-a` argument.

`ui`

Exports user interface settings.

`all`

Exports all of the available configuration areas. If you use this option, you must direct the output to a directory.

`-a <author_key>`

Export only the configuration items created by author with key `author_key` in a special XML format used for incremental import. The `nnmconfigimport.ovpl` script automatically detects this XML format. You do not need to use a special option when using the `nnmconfigimport.ovpl` script. This option is only available for configurations `author`, `customCorrelation`, `device`, `incident`, `menu`, and

`menuitem`. You can find available author keys by doing an export of authors. See the examples shown below.

`-f <output file or directory>`

Saves the output to the specified file or directory.

`-x <file_prefix>`

A file name prefix that is used to name files when the specified output is a directory. Files will be named <*prefix*>-<*area*>.xml.

EXAMPLES

`nnmconfigexport.ovpl -u myusername -p myadminpassword -c comm`

Exports the communication configuration to stdout.

`nnmconfigexport.ovpl -u myusername -p myadminpassword -c comm,disco -f /tmp -x my`

Exports the communication and discovery configurations to files named `/tmp/my-comm.xml` and `/tmp/my-disco.xml`.

`nnmconfigexport.ovpl -u myusername -p myadminpassword -c author`

Exports all authors with author key and label to stdout.

```
nnmconfigexport.ovpl -u myusername -p myadminpassword -c menuitem -a
com.mycorp.nnm.author -f /tmp/mycorpmenuitems.xml
```

Exports the menuitem configuration created by author with key `com.mycorp.nnm.author` to the `/tmp/mycorpmenuitems.xml` file.

AUTHOR

`nnmconfigexport.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnmconfigimport.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmconfigimport.ovpl — import the `nnmconfigexport.ovpl` XML output to the NNMi database.

SYNOPSIS

`nnmconfigimport.ovpl` -? | [-u <*`username`*> -p <*`password`*>] -f <*`input file or directory`*> [-x <*`file prefix`*>] [-memory <*`number of megabytes`*>] [-timeout <*`time in seconds`*>]

DESCRIPTION

`nnmconfigimport.ovpl` is a Perl script that enables you to import the output from the `nnmconfigexport.ovpl` script into the NNMi database.

NOTE: When performing an action import, you must restart the NNMi console (sign out, then sign in) for the changes to take effect.

Parameters

The `nnmconfigimport.ovpl` script supports the following options:

`-?`

Displays the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-f <input file or directory>`

Imports the configuration XML file, or, if you specify a directory, imports all of the files in that directory.

`-x <file prefix>`

The file prefix used to filter files to be imported when you use the `-f` option and specify a directory. Any files within the specified directory that are named <*`file prefix`*>-* will be imported, then sorted by dependencies.

`-memory <memory in megabytes>`

The amount of memory available for the `nnmconfigimport.ovpl` script to perform work. The default is 512 megabytes. Larger import files may require this value be set to 1024 or 2048. This option is not presented in the usage message since it is used by the `nnmconfigimport.ovpl` script and not passed to the actual import tool.

`-timeout <time in seconds>`

The amount of time available for the import of a particular file to complete. Some import types such as incidents might require a larger timeout value based on the amount of data. The default is 3600 (60 minutes). This option is not presented in the usage message since it is used by the `nnmconfigimport.ovpl` script, and not passed to the actual import tool.

NOTES

For a majority of areas `nnmconfigimport.ovpl` appends to the existing configuration. And in a few areas like Discovery, Communication, Monitoring, and Status, the existing configuration settings are replaced. Please see *Export/Import Behavior and Dependencies* in the Online Help for more information.

In order to add configuration elements, you must run the `nnmconfigexport.ovpl` script with a provided `author_key`.

The `nnmconfigimport.ovpl` script automatically detects if you ran the `nnmconfigexport.ovpl` script using a provided `author_key`, and adds configuration entries instead of replacing them.

The `Query Password` field of a RAMS configuration will only be valid when you import it into the same NNMi installation on the same system. If you import it into a different system, you will need to re-enter the `Query Password`.

Never edit the `nnmconfigexport.ovpl` output files before importing them.

EXAMPLES

```
nnmconfigimport.ovpl -u system -p openview -f /tmp/nnmconfig.xml
```

Imports the customized configuration in the `/tmp/nnmconfig.xml` file to the NNMi database. (You must provide an NNMi username and password. In this case, the username is `system` and the password is `openview`.)

AUTHOR

`nnmconfigimport.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnmconfigexport.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmconfigpoll.ovpl — poll a node for discovery information

SYNOPSIS

`nnmconfigpoll.ovpl` [-v] [-t *`timeout in secs`*] [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099] [-tenant <*`name`*>] *`node`*

DESCRIPTION

The `nnmconfigpoll.ovpl` script sends a request to the discovery service to poll the node for discovery information. The node must exist in the discovered topology. The *`node`* parameter you enter can be a node name in the topology or an IP address associated with a node.

Running the `nnmconfigpoll.ovpl` script starts a layer 2 connectivity analysis for the node. NNMi displays status messages for the layer 3 discovery information in the NNMi console as the discovery service polls the device.

The `nnmconfigpoll.ovpl` script polls for discovery information. To poll for status information, use the the `nnmstatuspoll.ovpl` script.

Parameters

The `nnmconfigpoll.ovpl` script recognizes the following parameters and options:

`-v`

Display verbose information about the discovery poll.

`-t <`*`timeout in secs`*`>`

The client waits *`timeout in secs`* seconds for a response.

`-u <`*`username`*`>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <`*`password`*`>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost <`*`serverName`*`>`

The server JNDI host; default is `localhost`.

`-jndiPort <port>`

The server JNDI port; default is `1099`.

`-tenant <name>`

The tenant group the node belongs to; must be used when the node name is not unique in the network such as using an IP address that has duplicates;no default.

EXAMPLES

The following examples show how to use the `nnmconfigpoll.ovpl` script to poll a node using different options.

Poll a node using its node name.

```
nnmconfigpoll.ovpl -u username -p password thisnode
```

Poll a node using its fully qualified node name.

```
nnmconfigpoll.ovpl thisnode.x.y.z
```

Poll a node using its IP address.

```
nnmconfigpoll.ovpl 10.97.247.129
```

Poll a node using its IP address and tenant name.

```
nnmconfigpoll.ovpl -tenant myDuplicateAddressesDomain 10.97.247.129
```

AUTHOR

`nnmconfigpoll.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_BIN%\nnmconfigpoll.ovpl`

UNIX: `$NNM_BIN/nnmconfigpoll.ovpl`

SEE ALSO

nnmstatuspoll.ovpl(1M).

Return to Reference Pages Index

Name

nnmconnedit.ovpl — make corrections to the L2 (layer 2) connection topology. Users can add and delete connections.

SYNOPSIS

`nnmconnedit.ovpl` -f *`corrections file`* -t [add|delete] [-help] [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

DESCRIPTION

Due to a variety of factors, NNMi L2 connection topology discovery can contain inaccuracies. The `nnmconnedit.ovpl` script provides a way for the user to add connections to or delete connections from NNMi. The administrator creates the *`corrections file`* and must follow the following structured XML format:

```
<connectionedits>
    <connection>
        <operation>add or delete</operation>
        <node>name, long name or IP address</node>
        <interface>ifName, ifAlias, ifDescr or ifIndex</interface>
        <node>name, long name or IP address</node>
        <interface>ifName, ifAlias, ifDescr or ifIndex</interface>
    </connection>
</connectionedits>
```

Where:

operation identifies whether the connection is to be added or deleted.

node is identified by its short name, long name (DNS name) or IP address.

interface is identified in order by `ifIndex`, `ifName`, `ifDescr`, or `ifAlias`. This value must be unique. Note that using `ifIndex` is discouraged due to the interface renumbering feature supported by some devices. For non-SNMP nodes, `ifAlias` or `ifDescr` are recommended.

For each connection element, there must be at least two nodes and two interfaces. The number of nodes and interfaces must be equal. Each node and interface pair is known as an endpoint, so a single connection element can have two or more endpoints specified. Multiple connection elements are permitted within a corrections file.

When adding a connection, each endpoint will be removed from any existing connection of which it may be a member, then added to the new connection. If there are more than two endpoints specified in the connection element, the connection will appear on maps as a shared-media connection symbol. If the connection element specifies a connection which already exists in the NNMi database, then nothing will be changed.

When deleting a connection, nothing will be changed unless a connection with the same set of endpoints already exists in the NNMi database. In that case, all of the specified endpoints will be left in a disconnected state. If network devices are reporting a connection, deleting that connection will only be

temporary. In this case, the deleted connection reappears in NNMi the next time NNMi discovers the nodes involved in the connection.

Parameters

The `nnmconnedit.ovpl` script recognizes the following parameters and options:

`-f` *`corrections file`*

Specifies the name of the file that contains the formatted connection add and delete directives.

`-t [add | delete]`

Generates a template file that can be used to create a corrections file. Specifying *add* will create an add operation template while specifying *delete* will generate a delete operation template.

`-u <`*`username`*`>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <`*`password`*`>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost <`*`serverName`*`>`

The server JNDI host; default is `localhost`.

`-jndiPort <`*`port`*`>`

The server JNDI port; default is `1099`.

`-help`

This option displays the script usage information.

EXAMPLES

Suppose that NNMi connection discovery was unable to find the L2 connection between two switches manufactured by different vendors. To remedy this, go to each device and get the node names and interface names that need to be connected. Next, create the add template file, `mychg.xml`, passing the `-t` option of the `nnmconnedit.ovpl` script. Next, edit the file, filling in the node and interface information. Save your changes in the `mychg.xml` file.

The following is an example of the `mychg.xml` (add template) file:

```
<connectionedits>
    <connection>
        <operation>add</operation>
        <node>nodeA.x.y.z</node>
        <interface>fa/09</interface>
        <node>nodeB.x.y.z</node>
```

```
        <interface>fa/05</interface>
    </connection>
</connectionedits>
```

Finally, run the `nnmconnedit.ovpl` script, passing the `mychg.xml` file with the `-f` option.

```
nnmconnedit.ovpl -f mychg.xml
```

Suppose that NNMi connection discovery creates an L2 connection where none should exist. To remedy this, get the node names and interfaces involved in the incorrect connection. Then create the `mychg.xml` (delete template) file, passing the `-t` option of the `nnmconnedit.ovpl` script. Next, edit the file by filling in the node and interface information. Save your changes in the `mychg.xml` file.

The following is an example of the `mychg.xml` (delete template) file:

```
<connectionedits>
    <connection>
        <operation>delete</operation>
        <node>nodeA.x.y.z</node>
        <interface>fa/09</interface>
        <node>nodeB.x.y.z</node>
        <interface>fa/05</interface>
    </connection>
</connectionedits>
```

Finally, run the `nnmconnedit.ovpl` script, passing the `mychg.xml` file with the `-f` option.

```
nnmconnedit.ovpl -u username -p password -f mychg.xml
```

AUTHOR

`nnmconnedit.ovpl` was developed by Hewlett-Packard Company.

FILES

None

See Also

None

Return to Reference Pages Index

Name

nnmdeleteattributes.ovpl — script to delete custom attributes from a comma separated values (CSV) file for both nodes and interfaces.

SYNOPSIS

`nnmdeleteattributes.ovpl` [-?] [-u <*username*> -p <*password*>] [-t <*object type*>] [-f <*csv_filename*>] [-s <*csv formatted line*>]

DESCRIPTION

`nnmdeleteattributes.ovpl` allows custom attributes to be deleted from a comma separated values (CSV) file, such as a .csv file from Microsoft™ Excel. This command is useful if you have previously created custom attributes that are no longer needed. This command will delete attributes from either nodes or interfaces. For nodes, once the attributes are deleted, any nodes in a node group formed by referencing those attributes will disappear from the node group.

Parameters

`nnmdeleteattributes.ovpl` supports the following options:

`-?`

Prints the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-t <object type>`

Supply the object type to load attributes on. Must be either "node" or "interface".

`-f <csv_filename>`

Supply the CSV file name (with path, e.g. /opt/tmp/mynodes.csv) that lists the custom attributes to delete.

`-s <csv formatted line>`

Supply a single CSV formatted line. Eliminates the need to create a file for a simple change

Syntax of Comma Separated File for Nodes

The CSV file you supply must have the following syntax for deleting attributes from nodes.

Empty lines are ignored.

Lines that being with the # character are ignored.

- Column 1(A) : Node DNS|IP Address

  Specify the DNS name of the node OR the IP address. This field is compulsory.

- Column 2(B) : Attribute Name

  The name of the custom attribute.

Additional attribute names may be specified on the same line or on a separate line with the same node DNS|IP Address.

Example lines:

192.168.1.1,Project,Service Type

192.168.1.1,Asset Tracking

192.168.2.2,Project,Service Type,Asset Tracking

Syntax of Comma Separated File for Interfaces

The CSV file you supply must have the following syntax for deleting attributes from interfaces.

Empty lines are ignored.

Lines that being with the # character are ignored.

- Column 1(A) : Node DNS|IP Address

  Specify the DNS name of the node OR the IP address. This field is compulsory.

- Column 2(B) : Interface Id

  Specify the identifier of the interface on the node specified in the previous field. The value may be the interface index, alias, name or description and is searched in this same order. All matching interfaces have the attribute(s) loaded. This field is compulsory.

- Column 3(C) : Attribute Name

  The name of the custom attribute.

Additional attribute names may be specified on the same line or on a separate line with the same node

DNS|IP Address and Interface Id.

Example lines:

192.168.1.1,1001,Project,Service Type

192.168.1.1,1001,Asset Tracking

192.168.2.2,A1,Project,Service Type,Asset Tracking

Use of Microsoft Excel

Microsoft Excel is a handy tool to create comma separated files, but .csv files do not maintain the Excel spreadsheet's column width, comments, etc. It is recommended that you store a nnmdeleteattributes.ovpl input file as a native .xls format, and then perform `File:Save As...` to create a .csv file. Then you can add Excel comments to the file, make columns wider, and you do not need to worry about escaping the comma character.

EXAMPLES

Sample CSV file contents for nodes:

```
192.168.2.2,Project,Service Type,Asset Tracking
```

To delete the Node custom attributes from a CSV file:

```
nnmdeleteattributes.ovpl -t node -u system -p myadminpasswd –f /tmp/test.csv
```

To delete the Node custom attributes from the command line:

```
nnmdeleteattributes.ovpl -t node -u system -p myadminpasswd -s "192.168.1.1,Project"
```

Sample CSV file contents for interfaces:

```
192.168.2.2,1001,Project,Service Type,Asset Tracking
```

To delete the Interface custom attributes from a CSV file:

```
nnmdeleteattributes.ovpl -t node -u system -p myadminpasswd –f /tmp/test.csv
```

To delete the Interface custom attributes from the command line:

```
nnmdeleteattributes.ovpl -t node -u system -p myadminpasswd -s "192.168.1.1,Project"
```

AUTHOR

`nnmdeleteattributes.ovpl` was developed by Hewlett-Packard Company.

FILES

```
$NNM_BIN/nnmdeleteattributes.ovpl
```

SEE ALSO

nnmloadattributes.ovpl(1M), nnmloadnodegroups.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmdeleteurlaction.ovpl — delete URL actions and menus for the supplied author key.

SYNOPSIS

`nnmdeleteurlaction.ovpl` [-h | -help | -?] <*`authorKey`*> [-u *`username`*] [-p *`password`*]

DESCRIPTION

The `nnmdeleteurlaction.ovpl` Perl script enables you to delete URL actions and menus by providing the author key to actions or menus.

NOTE: When using the `nnmdeleteurlaction.ovpl` script to delete an action or menu, you must restart the NNMi console (sign out; then sign back in) to implement the changes.

Use the `nnmdeleteurlaction.ovpl` script to remove (during uninstall) any actions or menus created during the installation of NNMi iSPIs or third party integrations.

Parameters

`nnmdeleteurlaction.ovpl` supports the following options:

`-? | -h | -help`

Displays the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

EXAMPLES

`nnmdeleteurlaction.ovpl com.hp.nms.author.nas`

Deletes the URL actions and menus that are created when the NAS integration is installed.

AUTHOR

`nnmdeleteurlaction.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnmconfigimport.ovpl(1M), nnmconfigexport.ovpl(1M).

Return to Reference Pages Index

Name

nnmdisableperfspi.ovpl — disables the optional HP iSPI for Performance Smart Plug In for use on the local NNM system

SYNOPSIS

`nnmdisableperfspi.ovpl`

DESCRIPTION

In order for NNM to stop collecting performance data and passing it to the Performance SPI for aggregation and reporting, several configuration changes must take place.

1. The SPI licenses must be uninstalled/disabled.
2. URL launch actions must be removed and will no longer show up in the NNM UI.
3. The Single Sign On service must be un-deployed from the NNM jboss server.

Disabling the license will prevent future data collection of performance metrics as well as disabling the ability to enable or disable performance monitoring of interfaces through monitoring configuration.

URL launch actions are removed from the NNM UI menu system. All NNM consoles that are currently running will need to exit and log back in before the action menu will reflect these changes. The URL actions that are removed are as follows: Reporting - Home Page, Reporting - Report Menu, Reporting - Data Explorer Live, Reporting - Chart Detail Live, Reporting - Path Health.

The ssoservlet that provides for single sign on between NNM and iSPI for Performance will be removed.

This script currently will not remove the user or shared drive that was setup during the `nnmenableperfspi.ovpl` script. These will need to be removed manually. Refer to the NNM iSPI for Performance documentation for additional information.

ENVIRONMENT

This script must be run with full administrative privileges and requires standard NNM environment variables to be configured.

AUTHOR

`nnmdisableperfspi.ovpl` was developed by Hewlett-Packard Company.

FILES

The `nnmdisableperfspi.ovpl` script removes the `ssoservlet.war` from the jboss structure.

SEE ALSO

nnmlicense.ovpl(1M) , nnmenableperfspi.ovpl(1M) .

Return to Reference Pages Index

Name

nnmdiscocfg.ovpl

SYNOPSIS

`nnmdiscocfg.ovpl` -autodisco rule=*`rulename`* rangetype=*`ignore|include`* [-f *`ipAddressRangeFile`* | -n *`ipAddressRanges`*] [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

`nnmdiscocfg.ovpl` -excludeipaddrs [-f *`ipAddressRangeFile`* | -n *`ipAddressRanges`*] [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

DESCRIPTION

The `nnmdiscocfg.ovpl` script permits the addition of ip address ranges to existing auto-discovery rules. The auto-discovery ip address ranges control how discovery finds devices in your network.

You can add discovery IP address range exclusion filters to prevent the creation of unwanted ip addresses in the NNMi topology. NNMi does not associate addresses matching the filters with nodes or interfaces and none of these nodes or interfaces will show up in the IP Address inventory. IP address range filters do not control how auto-discovery locates and identifies devices in your network.

Parameters

The `nnmdiscocfg.ovpl` script recognizes the following parameters and options:

`-autodisco rule=`*`ruleName`* `rangetype=ignore|include`

Add ip address ranges to an existing auto-disovery rule specified by *`ruleName`*. The ranges can be included in the rule by specifying rangetype=`include` or `ignore`.

`-excludeipaddrs`

Add ip address ranges to the discovery *Excluded IP Addresses* configuration.

`-f` *`ipAddressRangeFile`*

Specify a text file for the script to read that contains the IP address ranges.

`-n` *`ipAddressRanges`*

Specify IP address ranges to load directly from the command line, with each range separated by a space.

`-u <`*`username`*`>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost <serverName>`

The server JNDI host; default is `localhost`.

`-jndiPort <port>`

The server JNDI port; default is `1099`.

EXAMPLES

Add a list of IP address ranges to an existing auto-discovery rule:

```
 nnmdiscocfg.ovpl -autodisco rule=bld1floor2 rangetype=include -n 10.2.112.21-34
10.2.112.36 10.1.*.1-98
```

Add an IP address range to the IP Address exclusion configuration:

```
 nnmdiscocfg.ovpl -u username -p password -excludeipaddrs -n 198.2.*.117
```

Load IP address range from a file named privateNet1.txt located on your local files system in the tmp directory:

```
 nnmdiscocfg.ovpl -autodisco rule=bld1floor2 rangetype=exclude -f /tmp/lab3devices.txt
```

Load IP address range filters from a file named ignoreAddresses.txt located on your local files system in the tmp directory:

```
 nnmdiscocfg.ovpl -excludeipaddrs -f /tmp/ignoreAddresses.txt
```

AUTHOR

`nnmdiscocfg.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnmnoderediscover.ovpl(1M), nnm.properties(4)

Return to Reference Pages Index

Name

nnmenableperfspi.ovpl — enable the optional HP iSPI for Performance Smart Plug In for use on the local NNM system

SYNOPSIS

`nnmenableperfspi.ovpl`

DESCRIPTION

In order for NNM to begin collecting performance data and pass it to the Performance SPI for aggregation and reporting, several configuration changes must take place.

1. The SPI license must be enabled.
2. URL launch actions must be loaded into the NNM UI.
3. Drive space may need to be shared with the Performance SPI, possibly requiring a new operating system user to be configured on the NNM system.
4. The Single Sign On service must be deployed into the NNM jboss.

Enabling the license:

The license is initially enabled using an 'Instant On' setting which allows the Performance SPI to function for a limited period of time. The script achieves this by running the `nnmlicense.ovpl` command with the options `'PerfSPI -installIO'`

Loading URL Launch Actions:

Five separate launch actions are added into the NNM UI menu system. They are as follows: Reporting - Home Page, Reporting - Report Menu, Reporting - Data Explorer Live, Reporting - Chart Detail Live, Reporting - Path Health. The Path Health action is only available from the path view sections of the UI.

Sharing Drive Space (Required only when NNM and iSPI for Performance are on separate systems):

The directory where performance data is deposited by NNM is $NnmInstallDir/data/shared/perfSpi/datafiles. This must be shared with the Performance SPI system. When the Performance SPI and NNM co-exist on the same system there is no need to configure a network share of the directory structure. If NNM is installed on HPUX, Solaris, or Linux the script will attempt to configure either NFS or Samba for the correct network share. NFS should be used if NNM and the iSPI for Performance are both on Unix (HPUX, Solaris or Linux) systems. Samba should be used if NNM is on an HPUX, Solaris, or Linux system and the iSPI for Performance is on a Windows system. If both NNM and the iSPI for Performance are on Windows the script can configure a Windows shared folder for you. When Windows file sharing or Samba is used to share the performance data folder, there is a requirement to have use a shared OS account (identical username/password) on both the NNM machine (for exporting the share) and on the iSPI for Performance machine (for accessing the share). The `nnmenableperfspi.ovpl` script will establish this user account for you on the NNMi machine. Alternatively, the script offers you the opportunity to configure the operating

system user yourself.

Deploying the Single Sign On Service:

Users of NNM operate in an authenticated environment - they must provide a username and password in order to access the system. Those same credentials should be used to distinguish users of the Performance SPI system and to avoid double log in issues. In order for this to happen, NNM is configured as a 'trusted provider' for the Performance SPI. The SPI and NNM must be installed on systems which reside within the same network domain. The ssoservlet is part of the enabling apparatus for this functionality.

The following information will be required in order to complete the enablement:

- Fully qualified host name of the NNM system (Note: the enablement script will provide a default value which should work in most cases)

- Fully qualified host name of the SPI system - required only if the SPI will be installed on a remote server.

- Username and Password for enabling Windows or Samba file sharing - required only if the SPI will be installed on a remote *Windows* server.

ENVIRONMENT

This script must be run with full administrative privileges and required standard NNM environment variables to be configured.

AUTHOR

`nnmenableperfspi.ovpl` was developed by Hewlett-Packard Company.

FILES

The `nnmenableperfspi.ovpl` script loads URL actions into the NNM menu system. These actions are stored as XML files in the `newconfig` directory of the NNM installation. It also places `ssoservlet.war` into the jboss structure. The war file is stored within the `newconfig` directory prior to deployment.

SEE ALSO

nnmlicense.ovpl(1M) , nnmconfigimport.ovpl(1M) , nnmconfigexport.ovpl(1M) , nnmdisableperfspi.ovpl(1M) .

Return to Reference Pages Index

Name

nnmhealth.ovpl — Prints NNMi health information

SYNOPSIS

`nnmhealth.ovpl` [-u <*username*> -p <*password*>] (-print [quiet|brief|detailed|conclusions|verbose|agents|history] [-refresh] | -activate <*conclusions*> | -suppress <*conclusions*>)

DESCRIPTION

The `nnmhealth.ovpl` script prints out information about the internal health of NNMi. It supports several levels of output from no output, except for the return code (quiet), to the full health report (verbose).

In addition to viewing the current health of NNMi, you might use the `nnmhealth.ovpl` script to suppress or activate individual health conclusions. The administrator might suppress health conclusions if he or she is aware of the problem and does not want NNMi to report more warnings until the issue is resolved.

If the administrator modifies the suppressed list using the `nnmhealth.ovpl` script, the modifications take effect on the next health scan. The modifications to the suppressed list continue until the administrator activates them or until the adminstrator restarts NNMi.

Parameters

The `nnmhealth.ovpl` script supports the following options:

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-print <level>`

Prints information about the health of NNMi. The level can be one of the following: brief|conclusions|detailed||quiet|agents|verbose

If no level is specified the command will default to `brief`

`-print brief`: Prints the overall system status. The value can be one of the following:

Normal

Warning

Minor

Major

Critical

`-print conclusions`: Prints the active and surpressed conclusions.

`-print detailed`: Prints detailed system health information.

`-print agents`: Prints the registered agents.

`-print quiet`: Returns an integer value that represent the system status. The value can be one of the following:

0 - Normal

1 - Warning

2 - Minor

3 - Major

4 - Critical

`-print verbose [-filter AgentList]`: Prints verbose information for all registered agents. Optionally you can supply a list of comma separated agent names using the `-filter` option. The verbose output is for support use only.

`-refresh`

An optional argument to the `-print` command which will cause the montoring system to refresh its information before returning the report.

`-suppress <conclusions>`

Configures the supplied conclusions to be suppressed until NNMi is restarted or until the conclusion is activated again. Specify conclusions as a comma separated list. You can find active conclusions using the `-print conclusions` option.

Note that suppressing a conclusion does not imply it would otherwise be active.

`-activate <conclusions>`

Removes the supplied conclusions from the suppressed list. Specify the conclusions as a comma separated list. You can find suppressed conclusions available to be activated by using the `-print conclusions` option.

`-help`

Prints the usage statement.

EXAMPLES

`nnmhealth.ovpl -u username -p password -print brief`

Prints the overall status of NNMi.

`nnmhealth.ovpl -print brief -refresh`

Updates and prints the current status of NNMi. If no credentials are specified then the stored password for the logged in user will be used.

`nnmhealth.ovpl -u username -p password -print detailed`

Prints the current list of health warnings.

`nnmhealth.ovpl -u username -p password -print agents`

Prints the current list of registered agents that report health related information.

`nnmhealth.ovpl -u username -p password -print history`

Prints history information of registered health agents..

`nnmhealth.ovpl -u username -p password -suppress "SystemLowSwap,SystemLowSwapPercent"`

Configures NNMi to skip checking the health of swap space in abolute terms and also as a percentage.

`nnmhealth.ovpl -u username -p password -activate "SystemLowSwap"`

Configures NNMi to resume checking for the `SystemLowSwap` conclusion.

AUTHOR

`nnmhealth.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmicons.ovpl — NNMi UI Configuration

SYNOPSIS

`nnmicons.ovpl` -help

`nnmicons.ovpl` -list | -create (<*`iconSpec1,iconSpec2,...`*> | -file <*`file`*>) | -update (<*`iconSpec1,iconSpec2,...`*> | -file <*`file`*>) | -delete (<*`iconName1,iconName2,...`*> | -file <*`file`*>) [-preview] [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

DESCRIPTION

`nnmicons.ovpl` provides access to icons that are stored in the NNM database. The icons can be listed, created, update and deleted.

Parameters

`nnmicons.ovpl` supports the following commands:

`-list`

List the icons that are stored in the NNMi database

`-create (<iconSpec1,iconSpec2,...> | -file <file>)`

Create icons using either icon specifications or an input file.

`<iconSpec1,iconSpec2,...>`

comma separated list of icon specifications, where `iconSpecN` is of the form: `iconName:authorKey[:<iconImageSpec1>[:<iconImageSpec2>]]`

An `iconImageSpecN` is of the form, `size:path`. `size` is the size of the square image in pixels, and must either 16 or 32 `path` is the file path to the image file. The image file must be either a GIF, JPEG, or PNG, and the corresponding file suffix must be one .gif, .jpeg, .jpg, or .png

`-file`

Path to a file containing contain a list of `iconSpecs`, one per line. Blank lines and comment may also be included in the file. Comments are denoted by a '#' character at the beginning of a line.

`-update (<iconSpec1,iconSpec2,...> | -file <file>)`

Update icons using either icon specifications or an input file. If the icons do not exist, they will be created.

`<iconSpec1,iconSpec2,...>`

comma separated list of icon specifications, where `iconSpecN` is of the form: `iconName:authorKey:<iconImageSpec1>:<iconImageSpec2>:...`

An `iconImageSpec` is of the form, `size:path`. `size` is the size of the square image in pixels. `path` is the file path to the image file. The image file must be either a GIF, JPEG, or PNG, and the corresponding file suffix must be one .gif, .jpeg, .jpg, or .png

`-file`

Path to a file containing contain a list of `iconSpecs`, one per line. Blank lines and comment may also be included in the file. Comments are denoted by a '#' character at the beginning of a line.

`-delete (<iconName1,iconName2,...> | -file <file>)`

Delete icons using either icon names or an input file. If the icons do not exist, they will be ignored

`<iconName1,iconName2,...>`

`iconNameN` is the icon name associated with the icon.

`-file`

Path to a file containing contain a list of icon names, one per line. Blank lines and comment may also be included in the file. Comments are denoted by a '#' character at the beginning of a line.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost <hostname>`

The server jndi host; default is localhost.

`-jndiPort <port>`

The server jndi port; default is 1099.

EXAMPLES

List icons in NNMi database

```
nnmicons.ovpl -list
```

Create icons using icons specifications:

```
 nnmicons.ovpl -create
iconName1:com.customer.author:16:image16.gif:image32.gif,iconName2:com.customer.author:16:a
```

Update icons using a specification file:

```
 nnmicons.ovpl -update -f /tmp/iconSpecificationFile.txt
```

Delete icons using icon names:

```
 nnmicons.ovpl -delete iconName1,iconName2
```

AUTHOR

`nnmicons.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: *`install_dir`*`\bin\nnmicons.ovpl`

UNIX: `$NNM_BIN/nnmicons.ovpl`

SEE ALSO

Return to Reference Pages Index

Name

nnmincidentcfg.ovpl — create incident configurations from SNMP MIBs

SYNOPSIS

`nnmincidentcfg.ovpl` [[[-loadTraps *`mib_module_name`* [-authorLabel *`author_label`* -authorKey *`author_key`*]] | -deleteAuthor *`author_key`* | -deleteCategory *`category_key`* | -deleteFamily *`family_key`* | -disableAllTraps <*`true`*|*`false`*> [-u *`username`*] [-p *`password`*]]

DESCRIPTION

`nnmincidentcfg.ovpl` is used to create incident configurations for SNMP traps defined in a TRAP-TYPE or NOTIFICATION-TYPE macro in an SNMP MIB file. To load a MIB into NNMi for defining MIB Expressions or to display numeric SNMP Object Identifiers as text, use the `nnmloadmib.ovpl` command.

The created incident will have the following values, which can then be manually updated with the NNMi user interface:

1. Name will be set to the name of the trap/notification in the MIB file.

2. Oid will be set to the oid of the trap/notification in the MIB file.

3. Enable will be set to "true."

4. Category will be set to "Status."

5. Family will be set to "Node."

6. Severity will be set to "Normal."

7. Message Format will be set to the name of the incident configuration.

8. Description will be set to the trap/notification description in the MIB file.

The created incident can be accessed using the Incident Configuration view. This can be further customized as required.

`nnmincidentcfg.ovpl` supports a special annotation called *#SUMMARY*. The value for the #SUMMARY annotation will be used as Message Format value in the created incident configuration entry. This annotation is applied to the MIB file just after the trap description as a MIB comment. The following shows an example of this:

```
MyTrap TRAP-TYPE
ENTERPRISE   hp
VARIABLES {
    serverName, trapTime, volumeName, volumeNum
}
DESCRIPTION "The disk volume is out of space. Please consult
your sysop, and/or the proper manual."

--#SUMMARY  "Volume $1 on system $2 is out of space."
```

Parameters

`nnmincidentcfg.ovpl` supports the following options:

`-loadTraps <mib_module_name>`

Specify the MIB module name that has the trap definitions. `nnmincidentcfg.ovpl` parses trap/notification definitions (`TRAP-TYPE` or `NOTIFICATION-TYPE` macros) found in the MIB module, and creates incident configurations for each entry.

`-authorLabel <author_label>`

Specifies the label of the author for the target incident configurations. This is an optional parameter. If author label is specified, then author key must also be specified.

`-authorKey <author_key>`

Specifies the key of the author for the target incident configurations. This is an optional parameter. If author key is specified, then author label must also be specified. It is recommended that java packaging notation with your company's domain be used, such as `com.example.nnm.author`.

`-deleteAuthor <author_key>`

It may be desirable to delete an Author that is no longer being used by the incident configuration. This option may be used to delete an Author by specifying the Author key value as long as no configurations reference the Author object.

`-deleteCategory <category_key>`

It may be desirable to delete a Category that is no longer being used by the incident configuration. This option may be used to delete a Category by specifying the Category key value as long as no incidents or incident configurations reference the Category object.

`-deleteFamily <family_key>`

It may be desirable to delete a Family that is no longer being used by the incident configuration. This option may be used to delete a Family by specifying the Family key value as long as no incidents or incident configurations reference the Family object

`-disableAllTraps <true|false>`

If *`true`* all traps should be loaded as disabled in the incident configuration. The default value is *`false`*, meaning the incident configuration is enabled by default.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

EXAMPLES

```
nnmincidentcfg.ovpl -loadTraps "CISCO-VTP-MIB" -authorLabel "Cisco" -authorKey
com.example.cisco.nnm.author

nnmincidentcfg.ovpl -loadTraps "mpls"
```

AUTHOR

`nnmincidentcfg.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

RFC 2578 Structure of Management Information Version 2 (SMIv2)

RFCs 1155, 1212, 1215: SNMP Version 1 Structure of Management Information

RFCs 1902, 1903, 1904: SNMP Version 2 Structure of Management Information

nnmloadmib.ovpl(1M),

nnm.properties(4).

Return to Reference Pages Index

Name

nnmincidentcfgload.ovpl — load and validate formatted incident configuration files

SYNOPSIS

`nnmincidentcfgload.ovpl` { -load *`filename`* [-timeout *`timeout`*] [-memory *`memory`*] [-u <*`user name`*> -p <*`password`*>] [-jdniHost <*`host name`*> -jdniPort <*`port`*>] } { -validate *`filename`* [-timeout *`timeout`*] [-memory *`memory`*] [-u <*`user name`*> -p <*`password`*>] [-jdniHost <*`host name`*> -jdniPort <*`port`*>] } { -formats *`sourceFilename`* -formatd *`destinationFilename`* [-u <*`user name`*> -p <*`password`*>] [-jdniHost <*`host name`*> -jdniPort <*`port`*>] } { -expression *`expression`* }

DESCRIPTION

The `nnmincidentcfgload.ovpl` is used to load and validate incident configuration files. These configuration files must conform to the required `tag` format described in `nnmincidentcfg.format` for each configuration type.

Valid configuration types include the following:

```
*MgmtEventConfig
*PairwiseConfig
*RemoteNnmEventConfig
*SnmpTrapConfig
*SyslogMessageConfig
```

Incident configurations must first have been created using the NNMi console or be loaded into the database using either the `nnmtrapdload.ovpl` or `nnmincidentcfg.ovpl` commands.

Before using the `nnmincidentcfgload.ovpl` command, do one of the following:

```
*Use the nnmincidentcfgdump.ovpl command to create a formatted file. Edit the incident
configurations using the required format described in nnmincidentcfg.format.
*Use an editor to create the formatted file with all required tags and modifications
using the format requirements described in nnmincidentcfg.format.
```

When using the `nnmincidentcfgload.ovpl` command, note the following:

```
*To avoid errors, if you need to edit complex incident configurations, use the
nnmincidentcfg.ovpl command to create the formatted file rather than using a text editor.
*The content of the formatted file replaces the configurations that are stored in the
NNMi database.
*The nnmincidentcfgload.ovpl also re-formats the incident configuration file by inserting
white space where needed to clarify the format hierarchy. This re-formatted file does
not contain any original comments.
```

Parameters

`nnmincidentcfgload.ovpl` supports the following options:

`-load <`*`filename`*`>`

Loads the formatted incident configuration file you specify into the NNMi database. Invalid file formats are not loaded into the NNMi database. NNMi reports each validation error, including the line number and error.

`-validate <filename>`

Displays the line number and validation error for all errors encountered in the formatted incident configuration file you specify. It does not load the incident configurations into the NNMi database.

`-formats <sourceFilename>`

Formats the file you specify and writes the file to the filename specified using `-formatd <destinationFilename>`.

`-formatd <destinationFilename>`

Specifies the filename that will contain the formatted version of `-formats <sourceFilename>`.

`-expression <expression>`

Validates the specified expression.

`-timeout <timeout>`

Specifies the transaction time out in seconds that NNMi should use for the `nnmincidentdump.ovpl` command.

`-memory <memory>`

Specifies the maximum heap size in Megabytes (MB). The minimum <*memory*> value is 512 MB. The default <*memory*> value is 1536 MB.

`-u <username>`

You must supply the administrator (Windows) or root (UNIX) username to run the script. Required unless a nnm.properties(4) file exists.

`-p <password>`

You must supply the administrator (Windows) or root (UNIX) password to run the script. Required unless a nnm.properties(4) file exists.

`-jndiHost <host name>`

The server JNDI host. The default value is `localhost`.

`-jndiPort <port>`

The server JNDI port. The default value is `1099`.

EXAMPLES

Load an incident configuration file:

```
nnmincidentcfgload.ovpl -load dumped-config.tag
```

Validate an incident configuration file:

```
nnmincidentcfgload.ovpl -validate modified-config.tag
```

Reformat an incident configuration file:

```
nnmincidentcfgload.ovpl -formats custom.tag -formatd formatted-output.tag
```

Validate an expression string:

```
nnmincidentcfgload.ovpl -expression "ciaName like \"whatTimeIsIt\""
```

The following example configuraton file contains all of the required tags for an incident configuration of type `SnmpTrapConfig` :

```
*ConfigurationType=SnmpTrapConfig
        *Name MinimalistTrapConfig
        *Oid .1.3.4.5.6
        -Author
                -Key com.customer.author
        -Category
                -Key com.hp.nms.incident.category.Fault
        -Family
                -Key com.hp.nms.incident.family.Node
        -MessageFormat Custom message format
        -Severity MINOR
```

The following example adds an action to the SNMP Trap Incident configuration:

```
*ConfigurationType=SnmpTrapConfig
        *Name MinimalistTrapConfig
        *Oid .1.3.4.5.6
        -Author
                -Key com.customer.author
        -Category
                -Key com.hp.nms.incident.category.Fault
        -ActionConfiguration
            -Actions
                -Action
                -Command echo "hello" > /tmp/hello.test
                -CommandType SCRIPT_OR_EXECUTABLE
                -LifecycleState InProgress
        -Family
                -Key com.hp.nms.incident.family.Node
        -MessageFormat Custom message format
        -Severity MINOR
```

AUTHOR

`nnmincidentcfgload.ovpl` was developed by Hewlett-Packard Company.

FILES

NNMi provides example configuration files and a description of the valid formats in the following directory:

```
Windows: install_dir\examples\nnm\incidentcfg
UNIX: /opt/OV/examples/nnm/incidentcfg
```

SEE ALSO

nnmincidentcfgdump.ovpl (1M) .

nnmincidentcfg.format (4) .

Return to Reference Pages Index

Name

nnmincidentcfgdump.ovpl — copies incident configuration data from the database into a `tag` formatted file.

SYNOPSIS

`nnmincidentcfgdump.ovpl` { -dump <*`filename`*> [-authorKey <*`author(s)`*> | -name <*`names(s)`*> | -oid <*`oid pattern(s)`*> | -mib <*`mib name(s)`*>] [-type <*`type(s)`*>] [-timeout <*`timeout`*>] [-memory <*`memory`*>] [-u <*`user name`*> -p <*`password`*>] [-jdniHost <*`host name`*> -jdniPort <*`port`*>] } { -listAuthors [-u <*`user name`*> -p <*`password`*>] [-jdniHost <*`host name`*> -jdniPort <*`port`*>] }

DESCRIPTION

`nnmincidentcfgdump.ovpl` is used to copy incident configurations from the database into a `tag` formatted file. These configuration files can be edited and loaded back into the database by using `nnmincidentcfgload.ovpl`.

Valid configuration types include the following:

```
*MgmtEventConfig
*PairwiseConfig
*RemoteNnmEventConfig
*SnmpTrapConfig
*SyslogMessageConfig]
```

Incident configurations must first have been created using the NNMi console or be loaded into the database using either the `nnmtrapdload.ovpl` or `nnmincidentcfg.ovpl` commands.

Parameters

`nnmincidentcfgdump.ovpl` supports the following options:

`-dump <filename>`

Specifies the destination file to which the incident configurations will be copied. If this file exists on the filesystem before the configuration is copied, the user will be notified to delete it.

`-authorKey <author(s)>`

Specifies the author keys to include in the formatted configuration file. All configurations which are tied to the specified author keys are included. If this argument is not provided all author keys are included.

Cannot be used with the `-name, -oid, or -mib` parameters

`-name <name(s)>`

Specifies one or more incident configuration names to include in the formatted configuration file. All configurations with matching names will be included. If this argument is not provided all incident

configurations for the specified type are included.

Cannot be used with the `-authorKey, -oid, or -mib` parameters

.

`-oid <oid pattern(s)>`

Specifies one or more Object Identifier (OID) patterns to be included in the formatted configuration file. The OID Pattern must follow these syntax requirement:

```
*Can contain one wild card "*".
*Must start with ".".
*Can contain only numbers or wild card separated by "."
```

Cannot be used with the `-name, -authorKey, -mib, or -type` parameters

`-mib <mib names(s)>`

Specifies the MIB module that must be contained in an incident configuration to be included in the formatted configuration file. Note the following requirements for the MIB module name:

```
*Must already exist in the NNMi database.
*Must already have traps loaded in the NNMi database.
```

Cannot be used with the `-name, -authorKey, -oid, or -type` parameters

`-type <type(s)>`

Specifies one or more configuration types to include in the formatted configuration file. Only the configuration types that are specified are included. If this argument is not provided all configuration types are included.

Valid configuration types include the following:

NOTE: Configuration types are not case sensitive.

```
*MgmtEventConfig
*PairwiseConfig
*RemoteNnmEventConfig
*SnmpTrapConfig
*SyslogMessageConfig
```

Cannot be used with the `-oid, or -mib` parameters

`-timeout <timeout>`

Specifies the transaction time out in seconds that NNMi should use for the `nnmincidentdump.ovpl` command.

`-memory <memory>`

Specifies the maximum heap size in Megabytes (MB). The minimum <*memory*> value is 512 MB. The default <*memory*> value is 1536 MB.

`-u <username>`

You must supply the administrator (Windows) or root (UNIX) username to run the script. Required unless a nnm.properties(4) file exists.

`-p <password>`

You must supply the administrator (Windows) or root (UNIX) password to run the script. Required unless a nnm.properties(4) file exists.

`-jndiHost <host name>`

The server JNDI host. The default value is `localhost`.

`-jndiPort <port>`

The server JNDI port. The default value is `1099`.

EXAMPLES

Include the entire Events configuration:

```
nnmincidentcfgdump.ovpl -dump full-dump.tag
```

Include all Management and Remote Event configurations:

```
nnmincidentcfgdump.ovpl -dump type-dump.tag -type MgmtEventConfig
remoteNnmEventConfig
```

Include all NNMi and Customer Event configurations:

```
nnmincidentcfgdump.ovpl -dump nnm-and-customer-author-dump.tag -authorKey
com.hp.nms.author.nnm com.customer.author
```

Include all NNMi Management Event configurations:

```
nnmincidentcfgdump.ovpl -dump author-type-dump.tag -authorKey
com.hp.nms.author.nnm -type MgmtEventConfig
```

Include the following named configurations, OvApaIfDownUpPair and DuplicateCorrelation:

```
nnmincidentcfgdump.ovpl -dump names-dump.tag -name OvApaIfDownUpPair
DuplicateCorrelation
```

Include all trap configurations that were loaded by the CISCO-VTP-MIB to a tag file named ciscoVtpMib.tag:

```
nnmincidentcfgdump.ovpl -dump ciscoVtpMib.tag -mib CISCO-VTP-MIB
```

Include the SnmpLinkDown/Up trap configurations:

```
nnmincidentcfgdump.ovpl -dump snmpLinkDownAndUp.tag -oid .1.3.6.1.6.3.1.1.5.3
.1.3.6.1.6.3.1.1.5.4
```

Include all LinkDown trap configurations (This includes CiscoLinkDown):

```
nnmincidentcfgdump.ovpl -dump linkDownTraps.tag -oid .1.3.6.1.6.3.1.1.5.3.*
```

Include all CiscoSNMPTrap configurations:

```
nnmincidentcfgdump.ovpl -dump ciscoSnmpTraps.tag -oid
.1.3.6.1.6.3.1.1.5.*.1.3.6.1.4.1.9
```

List all available author key / label pairs

```
nnmincidentcfgdump.ovpl -listAuthors
```

AUTHOR

`nnmincidentcfgdumpovpl`was developed by Hewlett-Packard Company.

FILES

Configuration formats, and example configuration files for each configuration type are provided under the following directories.

```
Windows: install_dir\examples\nnm\incidentcfg:
UNIX: /opt/OV/examples/nnm/incidentcfg
```

`nnmincidentcfgdumpovpl`was developed by Hewlett-Packard Company.

SEE ALSO

nnmincidentcfgload.ovpl (1M) .

nnmincidentcfg.format (4) .

Return to Reference Pages Index

Name

nnmldap.ovpl — reload or view LDAP configuration.

SYNOPSIS

`nnmldap.ovpl` -reload | -info | -diagnose <*`username`*> | -encrypt <*`password`*>

DESCRIPTION

`nnmldap.ovpl` is a script that enables you to reload, view or diagnose changes to the Lightweight Directory Access Protocol (LDAP) sign-in configuration without restarting jboss.

Parameters

`nnmldap.ovpl` supports the following options:

`-info`

Displays the LDAP configuration, such as

```
Configuration=providerURL:"ldap://example.com:636/". Number of available Incident assignment users:0
```

`-reload`

Reloads the LDAP configuration.

`-diagnose <username>`

Verifies configuration in the ldap.properties file by attempting to access <*`username`*> in the Directory Service using the LDAP configuration parameters. This command will respond with information to help you diagnose LDAP configuration problems.

<*`username`*> must be a valid username in the Directory Service. It is the same name that is used in the NNMi console username prompt of the NNMi login screen.

`-encrypt <password>`

Encrypts the supplied LDAP bind password so that it can be safely stored in the `ldap.properties` file.

The output of this command should be copied into the `bindCredential` property in the `ldap.properties` file. Encrypted passwords start with the `{ENC}` prefix.

Encrypted passwords can only be decrypted by the same NNMi which created them. If the database is reset or the properties are copied to a new NNMi system then this command will need to be re-run to generate a new encrypted password. The exception to this is if you are using NNMi in an application failover or High Availability (HA) configuration. In application failover or HA configurations, the

encrypted password generated by the script is valid on both NNMi management servers (since the database is the same on both NNMi management servers).

EXAMPLES

`nnmldap.ovpl -info`

Returns the current LDAP configuration.

`nnmldap.ovpl -reload`

Reads modifications to the ldap.properties file (such as enabling or disabling LDAP).

`nnmldap.ovpl -diagnose <username>`

Shows configuration parameters for the ldap.properties file, and verifies that information can be extracted from the Directory Service.

`nnmldap.ovpl -encrypt password`

Returns the encrypted value of the supplied password string. For example:
`{ENC}Mgnb1w007XYYenHvAFf3dQ==`

AUTHOR

`nnmldap.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnmsso.ovpl(1M), ldap.properties(4).

Return to Reference Pages Index

Name

nnmlicense.ovpl — administer Network Node Manager licensing

SYNOPSIS

`nnmlicense.ovpl` [-h | -help]

`nnmlicense.ovpl` [*<PRODUCT>* [-nosync] [(-g|-gui)|(-install|-f *<lic_file>*]]

The `-f|-install` option installs license passwords into the licensing database.

DESCRIPTION

`nnmlicense.ovpl` provides license management for HP Network Node Manager (NNMi). License management includes the ability to retrieve license passwords from Hewlett-Packard, installing license passwords from a file, the removal of license passwords, and reporting valid licenses.

There are two steps to adding a license: updating the license database and notifying the running NNMi processes that new license information is available. The `-nosync` option performs the former but does not notify NNMi. If the `-nosync` option is not specified, the running NNMi system is automatically notified; it is not necessary to stop and restart NNMi.

The `-nosync` option allows you to perform multiple licensing tasks (e.g. removing a license and installing a replacement license). You then notify NNMi with the following:

```
nnmlicense.ovpl NNM
```

The `-g|-gui` option opens the license management GUI, which provides access to most license management capabilities:

Retrieve/Install License Key

Enables a license password to be retrieved from Hewlett-Packard.

Install/Restore License Key

Installs license passwords from a text file.

Request License Key through Email/Fax

Provides the ability to to request a license for systems that cannot transfer data to Hewlett-Packard.

Report License Key

Reports license information for the particular product.

Backup License File

Enables the backing up of a license file before installing or removing licenses.

Remove License Key

Removes license keys (usually temporary or emergency licenses provided by Hewlett-Packard).

Recover License Key

Adds back licenses that were previously removed.

The license management GUI enables multiple operations before updating the running NNMi system. For example, you can replace a license by removing the old license, and then adding a new one in its place.

Parameters

PRODUCT

The short name for the product being licensed.

`-nosync`

Prevents synchronization with the running NNMi system.

`-g|-gui`

Starts the license management GUI.

`-f|-install <`*`lic_file`*`>`

Install the license contained in the specified license file

EXAMPLES

To install a license password contained in a file named "license.txt", perform the following:

```
$NnmInstallDir/bin/nnmlicense.ovpl NNM -f license.txt
```

This will update the license database and notify NNMi of the licensing change. To avoid notifying NNMi, the `-nosync` option can be provided:

```
$NnmInstallDir/bin/nnmlicense.ovpl NNM -nosync -f license.txt
```

The following example demonstrates replacing a license with a new one. Suppose the customer is given a temporary license for the NNM product, with a capacity of 500 nodes. Sometime later this customer receives their official license. The temporary license must first be removed the permanent license is accepted. This can be done using the license management GUI. First remove the temporary license (Remove License Key), and then add the permanent license (Install/Restore License Key). The running NNMi system is updated when they exit the GUI.

AUTHOR

`nnmlicense.ovpl` was developed by Hewlett-Packard Company.

FILES

`$NnmInstallDir/misc/nms/lic/NNM.pdf`

Product definition file used by licensing.

`$NnmDataDir/shared/nnm/conf/licensing/NNM.bin`

Data file representing the license information that is consumed by `ovjboss`.

`<drive>:\Program Files (x86)\Common Files\Hewlett-Packard\HPOvLIC`

Program files and data files for licensing on Microsoft Windows systems.

`/opt/OV/HPOvLIC`

Program files for licensing on UNIX-based systems.

`/var/opt/OV/HPOvLIC`

Data files for licensing on UNIX-based systems.

SEE ALSO

Installation Guide for future details on licensing.

Return to Reference Pages Index

Name

nnmloadinterfacegroups.ovpl — script to load Interface Group definitions from a comma-separated values (CSV) file.

SYNOPSIS

`nnmloadinterfacegroups.ovpl` [-?] [-u <*username*> -p <*password*>] [-r true | false] -f <*csv_filename*>

DESCRIPTION

NOTE: This script will do as much validation as possible on the comma-separated values (CSV) file before injecting data in the NNMi data store.

The `nnmloadinterfacegroups.ovpl` script loads Interface Group definitions from a comma-separated values (CSV) file, such as a .csv file exported from Microsoft™ Excel. This script is useful if you have a large amount of interface data defined in an external data store, and you want to load it into the NNMi database as Interface Group definitions. After loading the contents of the .csv file into NNMi, you can use the Interface Group form to further refine the definition of each Interface Group.

Parameters

`nnmloadinterfacegroups.ovpl` supports the following options:

`-?`

Prints the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-r true | false`

Back up the existing Interface Group configuration before using this option.

`-r false` (the default setting) means if the Interface Group Name already exists in the NNMi database, the nnmloadinterfacegroups.ovpl command does not change the previous settings.

`-r true` means all the settings for any existing Interface Group with the same Name (column 1) are overwritten with the values in your CSV file. Caution: this is not a merge, it is a complete replacement

of that Interface Group configuration.

`-f <csv_filename>`

Enter the CSV file name and the path for the CSV file.

Syntax of Comma-Separated File

The CSV file you supply must have the following syntax:

Required. Column 1. You must provide a value for the Interface Group Name.

Optional. Columns 2-7 are optional. Leave any combination of these columns blank.

You do not need to add a comma to indicate the end of Column 7. No semicolon ";" is required at the end of a list of values in a CSV field within a column (between commas ",").

NNMi combines the results of all settings in the following manner:

1. NNMi first evaluates any ifType Filters (column 6). Interfaces must match at least one specification to belong to this Interface Group.

2. NNMi then evaluates any Additional Filters (column 7). Interfaces must also pass all Additional Filters specifications to belong to this Interface Group.

3. If a Node Group (column 5) is specified for this Interface Group, any interface in this group must be contained in a node that is a member of that Node Group.

Empty lines or lines starting with a # character are ignored as comments. Add the following comment to line 1 to make it easy to remember the syntax for each required column.

#InterfaceGroupName,[Notes],[AddtoFilterList],[AddtoPerformanceFilterList],[NodeGroupName],[ifType1;...],[AdditionalFilters]

- Column 1(A) : Interface Group Name

  Required. Specify the Name of the Interface Group you want to import. (This becomes the Name attribute value in the Interface Group form.)

- Column 2(B) : Notes

  Optional. Describe the Interface Group in your own terms. (This becomes the Notes field text in the Interface Group form.)

- Column 3(C) : Add to View Filter List

  Optional. Sets the Add to View Filter List field of the Interface Group form.

  `1` (the default setting) this Interface Group is available in the drop-down filter list when viewing tables, such as the All Interfaces table.

  `0` do not include this Interface Group in the view drop-down filter list.

  Recommendation: Set this value to `1` only for the most commonly used Interface Groups. Avoid too many Interface Groups or the view filter list is too long and difficult to use.

- Column 4(D) : Add to Performance Filter List

  Optional. Sets the NNMi iSPI Performance Add to Filter List field of the Interface Group form.

  `0` (the default setting) this Interface Group is not available as a filter in NNM iSPI Performance reports.

  `1` this Interface Group appears in the Optional Filters selection panel of the NNM iSPI Performance reports.

  Recommendation: Set this value to `1` only for Interface Groups that are needed as filters in NNM iSPI Performance reports.

- Column 5(E) : Node Group Name

  Optional. The specified Node Group serves as a filter for this Interface Group.

  Note: If you specify a Node Group, the Node Group must already exist in the NNMi database.

- Column 6(F) : ifType Filters

  Optional. Add ifType Filters settings, separated by semicolon ";" (After importing, these specifications appear on the ifType Filters tab of the Interface Group form.) Each ifType is identified by the ifType name.

  Provide the exact ifType name as it appears in the NNMi console.

  Example entries for ifType Filters:

  - ds0;ds0Bundle;ds1;ds1FDL;ds3;g703at2mb
  - ppp;pppMultilinkBundle;propPointToPointSerial;slip
  - ethernetCsmacd

- Column 7(G) : Additional Filters

  Optional. Specify additional filter expressions used to further define the interfaces to be included in an Interface Group. The format of additional filters is:

  1. Define a filter condition operator and its associated filter conditions within a matching set of parentheses.

  2. Define a filter by specifying the filter attribute followed by a filter operator and then finally by the filter value.

  All filter attributes and operators that are available in the interface group form are supported. Multiple filter values can be specified by using a ':' to separate them while multiple filters for a filter condition operator will be separated by a ';'. A space is used to separate a filter attribute and the filter operator and a space is also used to separate a filter operator and the filter value.

  Filter Attributes:

- ifAlias
- ifDesc
- ifIndex
- ifName
- ifSpeed
- hostedOn
- ipAddress
- isSnmpInterface
- sysOidInterface
- devCategoryInterface
- devVendorInterface
- devFamilyInterface
- customAttrName
- customAttrValue
- capability
- vlanid
- vlanName
- ipPhysAddress
- configuredDuplexSetting

Filter Operators:

- !=
- >
- >=
- <
- <=
- =
- between
- in

- is_not_null
- is_null
- like
- not_between
- not_in
- not_like

Filter Condition Operators:

- AND
- OR
- NOT
- EXISTS
- NOT_EXISTS

Examples:

- `(AND hostedOn like *.mycompany.com (OR (EXISTS (AND customAttrName = circuit; customAttrValue = 12) ) (EXISTS (AND customAttrName = circuit; customAttrValue = 15) ) ) )`
- `(AND hostedOn like *.mycompany.com (EXISTS (AND customAttrName = circuit; customAttrValue in 12:15) ) )`
- `hostedOn like *.mycompany.com`
- `ifAlias = " Alias with leading and trailing spaces "`
- `ifAlias = Alias with embedded \"double quotes\"`

Note

When entering filter values, avoid using the special characters '"', '(', ')', ':' and ';'. If you need to use these special characters then escape them with a '\'. For example:

- 'circuit:57' must be entered as 'circuit\:57'
- 'circuit(57)' must be entered as 'circuit\(57\)'
- 'circuit"57"' must be entered as 'circuit\"57\"'
- 'circuit;57' must be entered as 'circuit\;57'
- 'circuit\:57' must be entered as 'circuit\\\:57'

Use of Microsoft Excel

Microsoft Excel is a useful tool to create comma-separated files, but .csv files do not maintain their column width, comments, and other spreadsheet options. HP recommends that you store a `nnmloadinterfacegroups.ovpl` input file as a native .xls format, then complete a `File:Save As...` command to create a .csv file. You can then add Excel comments to the file, make columns wider, and do not need to escape the comma character.

Note that typing a comma in a line after a leading # in Microsoft Excel generates a non-commented entry when the .xls file is saved as a .csv file (creating a Interface Group with a Name starting with the # character).

EXAMPLES

Sample CSV file contents:

```
Point to Point Interfaces,Point to Point Interfaces are usually associated with dial-up.,1,1,,ppp;pppMultilinkBundle;propPointToPointSerial;slip
```

To load the Interface Groups from a CSV file without overwriting any existing Interface Group that matches a Name defined in column 1 of your CSV file:

```
nnmloadinterfacegroups.ovpl -u system -p myadminpasswd -f /tmp/test.csv
```

To load the Interface Groups from a CSV file, and overwrite any existing Interface Group that matches a Name defined in column 1 of your CSV file (Caution: this is not a merge, it is a complete replacement of that matching Interface Group's configuration):

```
nnmloadinterfacegroups.ovpl -u system -p myadminpasswd -r true -f /tmp/test.csv
```

AUTHOR

`nnmloadinterfacegroups.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_BIN%\nnmloadinterfacegroups.ovpl`

UNIX: `$NNM_BIN/nnmloadinterfacegroups.ovpl`

SEE ALSO

nnmconfigimport.ovpl(1M), nnmloadnodegroups.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmloadipmappings.ovpl — load overlapping IP address information

SYNOPSIS

`nnmloadipmappings.ovpl` -f *`mapping file`* [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

DESCRIPTION

`nnmloadipmappings.ovpl` allows customer to load IP address mappings configured in static NAT [RFC2663] environment from a text file. The loaded mappings will be populated into the corresponding IP Address inventory.

The -f option accepts a file with a single entry specified per line. Each line has the following format:

`Tenant Name`,"*`Public IP Address`*","*`Private IP Address`*"

Where:

`Tenant Name` = The name of a tenant. *`Public IP Address`* = A specific NATed IPv4 address exposed to the ourside network. *`Private IP Address`* = A specific internal IPv4 address corresponding to the NATed public IP address.

Comments can be deliminated with a `#` character.

Note that one public IP address can only map to one private IP address in a tenant. Same for the private IP address in a tenant, it only can map to one public IP address. However, multiple mappings on a device is supported.

Parameters

The `nnmloadipmappings.ovpl` command recognizes the following parameters and options:

`-f` *`mapping file`*

Specify a text file to read the IP Address mappings from.

`-u` <*`username`*>

You must supply the administrator (Windows) or root (UNIX) username to run the script. Required unless a nnm.properties(4) file exists.

`-p` <*`password`*>

You must supply the administrator (Windows) or root (UNIX) password to run the script. Required unless a nnm.properties(4) file exists.

`-jndiHost <serverName>`

The server JNDI host; default is `localhost`.

`-jndiPort <port>`

The server JNDI port; default is `1099`.

EXAMPLES

Load IP address mappings from a file name that is named Tenant1Mappings.txt:

```
nnmloadipmappings.ovpl -f /tmp/Tenant1Mappings.txt
```

Load IP address mappings from a file name that is named Tenant2Mappings.txt with username and password:

```
nnmloadipmappings.ovpl -u username -p password -f /tmp/Tenant2Mappings.txt
```

AUTHOR

`nnmloadipmappings.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: *`install_dir`*`\bin\nnmloadipmappings.ovpl`

UNIX: `$NNM_BIN/nnmloadipmappings.ovpl`

Name

nnmloadattributes.ovpl — script to load custom attributes from a comma separated values (CSV) file for nodes and interfaces.

SYNOPSIS

`nnmloadattributes.ovpl` [-?] [-u <*username*> -p <*password*>] [-t <*object type*>] [-f <*csv_filename*>] [-r <*true | false*>] [-s <*csv formatted line*>]

DESCRIPTION

NOTE: This script will not validate comma-separated values (CSV) files before injecting data in the NNMi data store.

`nnmloadattributes.ovpl` allows custom attributes to be loaded from a comma separated values (CSV) file, such as a .csv file from Microsoft™ Excel. This command is useful if you have a large number of nodes or interfaces that are defined in an external datastore, and you would like to load these attributes into NNM. For nodes, after loading into NNM you can use the Node Group forms to group nodes according to their custom attributes.

Parameters

`nnmloadattributes.ovpl` supports the following options:

`-?`

Prints the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-r <true | false>`

Existing attribute values are not changed unless this argument is provided with the value of true.

`-t <object type>`

Supply the object type to load attributes on. Must be either "node" or "interface".

`-f <csv_filename>`

Supply the CSV file name (with path, e.g. /opt/tmp/mynodes.csv) from where you want to import the custom attributes.

`-s <csv formatted line>`

Supply a single CSV formatted line. Eliminates the need to create a file for a simple change

Syntax of Comma Separated File For Node Attributes

The CSV file you supply must have the following syntax for adding attributes on nodes. See below for the syntax for interface attributes.

Empty lines are ignored.

Lines that being with the # character are ignored.

- Column 1(A) : Node DNS|IP Address

  Specify the DNS name of the node OR the IP address. This field is compulsory.

- Column 2(B) : Attribute Name

  The name of the custom attribute.

- Column 3(C) : Attribute Value

  The value of the custom attribute.

Additional attribute name and value pairs may be specified on the same line or on a separate line with the same node DNS|IP Address.

Example lines:

192.168.1.1,Location,Building Five Upper,Service Type,eCommerce

192.168.1.1,Asset Tracking,N1234

192.168.2.2,Location,Fort Collins,Service Type,IT,Asset Tracking,F4321

Syntax of Comma Separated File For Interface Attributes

The CSV file you supply must have the following syntax for adding attributes on interfaces..

Empty lines are ignored.

Lines that being with the # character are ignored.

- Column 1(A) : Node DNS|IP Address

  Specify the DNS name of the node OR the IP address. This field is compulsory.

- Column 2(B) : Interface Id

  Specify the identifier of the interface on the node specified in the previous field. The value may be the interface index, alias, name or description and is searched in this same order. All matching interfaces have the attribute(s) loaded. This field is compulsory.

- Column 3(C) : Attribute Name

  The name of the custom attribute.

- Column 4(D) : Attribute Value

  The value of the custom attribute.

Additional attribute name and value pairs may be specified on the same line or on a separate line with the same node DNS|IP Address and Interface Id.

Example lines:

192.168.1.1,1001,Location,Building Five Upper,Service Type,eCommerce

192.168.1.1,1001,Asset Tracking,N1234

192.168.2.2,A1,Location,Fort Collins,Service Type,IT,Asset Tracking,F4321

Use of Microsoft Excel

Microsoft Excel is a handy tool to create comma separated files, but .csv files do not maintain the Excel spreadsheet's column width, comments, etc. It is recommended that you store a nnmloadattributes.ovpl input file as a native .xls format, and then perform `File:Save As...` to create a .csv file. Then you can add Excel comments to the file, make columns wider, and you do not need to worry about escaping the comma character.

EXAMPLES

Sample node CSV file contents:

```
192.168.2.2,Location,Fort Collins,Service Type,IT,Asset Tracking,F4321
```

To load the Node custom attributes from a CSV file overwriting existing values:

```
nnmloadattributes.ovpl -t node -u system -p myadminpasswd –f /tmp/test.csv -r true
```

To load the Node custom attributes from the command line:

```
nnmloadattributes.ovpl -t node -u system -p myadminpasswd –s "192.168.1.1,Project,IT
Update of Building Five"
```

Sample interface CSV file contents:

```
192.168.2.2,A1,Location,Fort Collins,Service Type,IT,Asset Tracking,F4321
```

To load the Interface custom attributes from a CSV file overwriting existing values:

```
nnmloadattributes.ovpl -t interface -u system -p myadminpasswd –f /tmp/test.csv -r true
```

To load the Interface custom attributes from the command line:

```
nnmloadattributes.ovpl -t interface -u system -p myadminpasswd –s
"192.168.1.1,1001,Project,IT Update of Building Five"
```

AUTHOR

`nnmloadattributes.ovpl` was developed by Hewlett-Packard Company.

FILES

`$NNM_BIN/nnmloadattributes.ovpl`

SEE ALSO

nnmdeleteattributes.ovpl(1M), nnmloadnodegroups.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmloadmib.ovpl — load and unload SNMP MIBs

SYNOPSIS

`nnmloadmib.ovpl` [[-load *`mib-file`*] [-unload *`mib module`*[:*`mib module...`*]] [-list] [-u *`username`*] [-p *`password`*] [-jndiHost *`hostname`*] [-jndiPort *`port`* Default is 1099]]

DESCRIPTION

`nnmloadmib.ovpl` is a script that loads an SNMP Management Information Base (MIB) in the Internet Structure of Management Information (SMI) format into NNMi. NNMi applications use this MIB information when converting SNMP Object Identifiers (OIDs) from numeric to human readable text. Load any new MIB information into NNMi before using the NNMi console to create a MIB expression that relies on the newly loaded MIB information. NNMi supports SMI Version 1 (RFC1155, 1212, 1215) and SMI Version 2 (RFC2578) SMI formats.

The `nnmloadmib.ovpl` script compiles and loads MIB modules for use by NNMi applications and stores the resulting information in the NNMi database.

To load corresponding NNMi Incident Configuration for TRAP-TYPE and NOTIFICATION-TYPE macros from a MIB, use the `nnmincidentcfg.ovpl` script after using the `nnmloadmib.ovpl` script to load the MIB into the NNMi MIB database.

Parameters

The `nnmloadmib.ovpl` script supports the following parameters:

`-load` *`mib-file`*

Load the contents of the *`mib-file`* file into the MIB database.

Note: HP recommends copying the *`mib-file`* file to the `$NNM_DATA/shared/nnm/user-snmp-mibs` (UNIX) or `%NNM_DATA%\shared\nnm\user-snmp-mibs` (Windows) directory (or a child directory) before running the **nnmloadmib.ovpl** script. Copying the *`mib-file`* file in this way enables the **Actions > Display MIB File** menu and the `-list` option to locate the original MIB file.

`-unload` *`mib module`*[:*`mib module...`*]

Unload *`the list of MIB modules`* from the MIB database, where *`mib module`* is the name of a MIB module that has been loaded into the MIB database.

`-list`

Lists the MIBs loaded in the database.

This option does not require the user to provide a username and password.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost<jndiHost>` (optional)

The hostname of the server running the jboss application server. If you do not specify a hostname, the `nnmcommload.ovpl` script uses `localhost` as the default value.

`-jndiPort<jndiPort>` (optional)

The jboss application server port. If you do not specify this port, the `nnmcommload.ovpl` script uses 1099 as the default value.

Loading/Unloading Validation

When loading and unloading MIBs all of the existing MIBs that have been loaded are parsed to ensure that the load/unload operation will pass given the current state. This may result in warnings being displayed (such as overlapping OIDs) that my be unrelated to the given MIB being loaded or unloaded. As long as a successful status is returned from the command line invocation, all is well and these warnings can be ignored.

Syntax of MIB Files

Most of the relevant syntax for MIB files is described in RFC documents. See the "SEE ALSO" section of this reference page (and the UNIX manpage).

Diagnostics

The `nnmloadmib.ovpl` script returns the following exit codes:

`0`

The script ran successfully.

`1`

Invalid command-line usage.

`20`

An unexpected exception occurred after running the script.

`21`

You were not permitted to run the script with the credentials you supplied.

22

The script detected a syntax error in the MIB file, or there was a failure to load the MIB due to a service failure.

23

You used an illegal argument - typically you typed an option such as `-load` without including the required file name.

24

The script could not communicate with the mib loader service running in NNMi.

25

You did not supply any arguments. This script requires you to supply several arguments

27

NNMi is available, but the mib loader service is not resolvable.

30

The list operation failed due to database issues.

When the `nnmloadmib.ovpl` command fails, it will display a descriptive error message to help diagnose and fix potential problems with the MIB. All error messages have a similar format:

```
SEVERITY: MESSAGE FILENAME:LINE_NUMBER:COLUMN_NUMBER: DETAIL_MESSAGE
```

Below are some common failures and recommended fixes for each.

```
ERROR: Cannot find symbol file:///tmp/CHECKPOINT-MIB.mib:2620:16:cpvTNlMonCurrAddr
```

The symbol name listed was either not found as a declaration in the MIB being loaded, or it is not listed as an import at the top of the MIB definition. This could be caused by a typo in the symbol name or a missing import delaration.

```
ERROR: Found symbol file:///tmp/CHECKPOINT-MIB.mib:3509:27:routingDest but expected a
class org.jsmiparser.smi.SmiType instead of class org.jsmiparser.smi.SmiVariable
```

The symbol name listed is not of the expected type. In this case, an SMI type was expected, but a MIB variable name was provided instead. The solution in this particular case was to specify the correct SMI type, which was IPAddress.

```
ERROR: Cannot find module file:///tmp/rfc1472-PPP-SEC-MIB.mib:9:26:PPP-LCP-MIB
```

This error indicates that the specified MIB module, which is listed as an import, has not been loaded, and the parser cannot follow the import as a result. The solution is to load the referenced MIB first.

```
ERROR: Parse error: unexpected token: --#
```

NNMi supports some custom trap message formatting information that can be specified in a TRAP-TYPE or NOTIFICATION-TYPE macro definition. The following are valid values:

```
--#TYPE
--#SUMMARY
--#ARGUMENTS
--#SEVERITY
--#GENERIC
--#CATEGORY
--#SOURCE_ID
--#TIMEINDEX
--#HELP
--#HELPTAG
--#STATE
```

This error indicates you have invalid keyword following the --#, or a sequence of keywords that NNMi does not expect. To correct this problem, remove any --# entries that do not correspond to the above list, or add an extra comment character sequence (--) to the beginning of the line.

EXAMPLES

To load the `$NNM_DATA/shared/nnm/user-snmp-mibs/corp.mib` MIB file, run the `nnmloadmib.ovpl` script as follows:

```
nnmloadmib.ovpl -load $NNM_DATA/shared/nnm/user-snmp-mibs/corp.mib -u user -p password
```

To list the loaded MIBs, run the `nnmloadmib.ovpl` script as follows::

```
nnmloadmib.ovpl -list -u user -p password
```

AUTHOR

`nnmloadmib.ovpl` was developed by Carnegie-Mellon University and Hewlett-Packard Company.

FILES

Windows: *%NnmInstallDir%*`\misc\nnm\snmp-mibs\*`

Windows: *%NnmDataDir%*`\shared\nnm\user-snmp-mibs/*`

UNIX:*$NnmInstallDir*`/misc/nnm/snmp-mibs/*`

UNIX:*$NnmDataDir*`/shared/nnm/user-snmp-mibs/*`

SEE ALSO

RFC 2578 Structure of Management Information Version 2 (SMIv2)

RFCs 1155, 1212, 1215: SNMP Version 1 Structure of Management Information

RFCs 1902, 1903, 1904: SNMP Version 2 Structure of Management Information

nnmincidentcfg.ovpl(1M),

nnmsnmpwalk.ovpl(1M).

Return to Reference Pages Index

Name

nnmloadnodegroups.ovpl — script to load Node Group definitions from a comma-separated values (CSV) file.

SYNOPSIS

`nnmloadnodegroups.ovpl` [-?] [-u <*username*> -p <*password*>] [-r true | false] -f <*csv_filename*>

DESCRIPTION

NOTE: This script will not validate comma-separated values (CSV) files before injecting data in the NNMi data store.

The `nnmloadnodegroups.ovpl` script loads Node Group definitions from a comma-separated values (CSV) file, such as a .csv file exported from Microsoft™ Excel. This script is useful if you have a large amount of node data defined in an external datastore, and you want to load it into the NNMi database as a starting point for Node Group definitions. After loading the contents of the .csv file into NNMi, you can use the Node Group form to further refine the definition of each Node Group.

The following settings cannot be set in the CSV file. You must import the Node Group and then use the Node Group form to modify these default settings:

- `Calculate Status = true` (NNMi calculates the Node Group status)

Parameters

`nnmloadnodegroups.ovpl` supports the following options:

`-?`

Prints the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-r true | false`

Back up the existing Node Group configuration before using this option.

`-r false` (the default setting) means if the Node Group Name already exists in the NNMi database, the nnmloadnodegroups.ovpl command does not change the previous settings.

`-r true` means all the settings for any existing Node Group with the same Name (column 1) are overwritten with the values in your CSV file. Caution: this is not a merge, it is a complete replacement of that Node Group configuration.

NOTE: When using the replace option with groups in a hierarchy it is required that all child groups are included in the same CSV file.

`-f <csv_filename>`

Enter the CSV file name and the path for the CSV file.

Syntax of Comma-Separated File

The CSV file you supply must have the following syntax:

Required. Column 1. You must provide a value for Node Group Name.

Optional. Columns 2-15 are optional. Leave any combination of these columns blank.

You do not need to add a comma to indicate the end of Column 15. No semicolon ";" is required at the end of the last entry within a column (between commas ",").

NNMi combines the results of all settings in the following manner:

1. NNMi first evaluates any Device Filters (column 5). Nodes must match at least one specification to belong to this Node Group.

2. NNMi then evaluates any Additional Filters (columns 7-14). Nodes must also pass all Additional Filters specifications to belong to this Node Group. Note: After importing, the Node Group form's Additional Filters tab displays the combined columns 7-14 settings. You can change the default Boolean logic using the Node Group form's Additional Filters Editor.

3. Any Additional Nodes specified (column 6) are always included in the Node Group, regardless of any filters.

4. Any Child Node Groups (column 4) results are treated the same as Additional Nodes.

Empty lines or lines starting with a # character are ignored as comments. Add the following comment to line 1 to make it easy to remember the syntax for each required column.

#[NodeGroupName],[Notes],[AddtoFilterList],[ChildNodeGroup:0/1;...],DeviceFilter[Category1:Vendor1:Family1:Profile1;...],AdditionalNodes[Fully-Qaul-hostname;...],AdditionalFilters > [hostname;...],[hostedIPAddress;...],[mgmtIPAddress;...],[customAttrName/customAttrValue;...],[capability;...]

- Column 1(A) : Node Group Name

  Required. Specify the Name of the Node Group you want to import. (This becomes the Name attribute value in the Node Group form.)

- Column 2(B) : Notes

  Optional. Describe the Node Group in your own terms. (This becomes the Notes field text in the Node Group form.)

- Column 3(C) : Add to View Filter List

  Optional. Sets the Add to View Filter List field of the Node Group form.

  `1` (the default setting) means this Node Group is available in the drop-down filter list when viewing tables, such as the All Nodes table.

  `0` means do not include this Node Group in the view drop-down filter list.

  Recommendation: Set this value to `1` only for top-level or most commonly used Node Groups. Avoid too many Node Groups or the view filter list is too long and difficult to use.

- Column 4(D) : Child Node Groups

  Optional. Specify a list of child Node Groups for this Node Group, separated by semicolon ";" (After importing, these specifications appear on the Child Node Groups tab of the Node Group form.) Note: If you are configuring Child Node Groups, the specified child Node Group must either already exist in the NNMi database or be defined in the same CSV file.

  Example: `ChildNodeGroup1:1[;ChildNodeGroup2:0;...]`

  `0` (the default setting) means that child Node Group is shown as a Node Group icon in maps of the parent Node Group.

  `1` means expand the child Node Group in a map of the parent Node Group. This option displays all nodes as if they were defined within the parent Node Group.

  Valid entries for Child Node Groups are:

  - computers:1
  - computers:0
  - computers:
  - computers:;printers:1

- Column 5(E) : Device Filters

  Optional. Add Device Filters settings, separated by semicolon ";" (After importing, these specifications appear on the Device Filters tab of the Node Group form.) Each filter specification consists of 4 optional colon-separated parts in the following format:

  ```
  Category1:Vendor1:Family1:Profile1[;Category2:Vendor2:Family2:Profile2 ...]
  ```

  Provide the exact specification from the device's MIB file (not the text string displayed in the NNMi console's Device Profile form).

  To match more filters, you may omit portions of a filter specification. For example, if you want to match any family for Category1 and Vendor1, add an entry such as the following:

  ```
  Category1:Vendor1::
  ```

  To leave family unspecified for filter1 and family and profile unspecified for filter2:

```
Category1:vendor1::profile1;Category2:vendor2::;
```

Valid example entries for device profile:

- com.hp.ov.nms.devices.printer:com.hp.ov.nms.devices.hewlettpackard::.1.3.6.1.4.1.9.1.380
- com.mycomp.ov.nms.devices.printer:com.hp.ov.nms.devices.mycompanyname::
- com.hp.ov.nms.devices.printer:::
- :::.1.3.6.1.4.1.9.1.380

- Column 6(F) : Additional Nodes

  Optional. Specify a list of specific node hostnames you want added to this Node Group, separated by a semicolon ";" (After importing, these specifications appear on the Additional Nodes tab of the Node Group form.) The hostnames you provide must be the current value of the fully-qualified, case-sensitive Hostname attribute as it appears on the Node form.

  For example `hostname1.x.y.z;hostname2.x.y.z;hostname3.x.y.z`

- Column 7(G) : Additional Filters "sysName" code (Hostname Wildcards)

  Optional. List the hostname wildcards separated by semicolon ";" (equivalent to the Operator "="). If you need other Operators, use the Node Group form after importing (these specifications appear on the Additional Filters tab of the Node Group form).

  For example: `*.cnd.hp.com;*snmp.hp.com`

- Column 8(H) : Additional Filters "hostedIPAddress" code (Hosted IP Address Ranges)

  Optional. List the hosted IP address ranges separated by semicolon ";" (equivalent to the Operator "=") If you need other Operators, use the Node Group form after importing (these specifications appear on the Additional Filters tab of the Node Group form). Ranges have a lower and an upper address, separated by a dash. The addresses are inclusive. To include a single IP address, use the same value for the lower and upper address values. Note that if any address on a node matches this range, the node will be included in the Node Group.

  Valid example: `10.20.30.1-10.20.30.254;192.168.177.1-192.168.180.254;1.1.1.1-1.1.1.1`

- Column 9(I) : Additional Filters "mgmtIPAddress" code (Management Address Ranges)

  Optional. List the management Address ranges separated by semicolon ";" (equivalent to the Operator "=") Ranges are in the same format as hosted IP address ranges. If you need other Operators, use the Node Group form after importing (these specifications appear on the Additional Filters tab of the Node Group form). Note that Spiral Discovery only creates management IP Addresses on nodes that support SNMP. See the online help for the Node form's Management Address field for more information about how Spiral Discovery selects the Management Address.

- Column 10(J) : Additional Filters "customAttrName:customAttrValue" codes (Custom Node Attributes)

  Optional. List the custom attributes assigned to nodes as follows: `"custom attribute name" operator "custom attribute value"[;...]` and note the name and the value must be surrounded by quotes. After importing, these Custom Node Attribute specifications appear on the Additional Filters tab of the Node Group form.

  Valid values for operator are as follows:

  `=, !=, like, not like, between, not between, >, >=, <, <=, is null, is not null` (If you need other values, use the Node Group form after importing.) The operators `is null` and `is not null` do not have a value, for example, `"my attribute" is not null`. The values for `between` and `not between` are specified as `x AND y` (for example, `"my attribute" between "100 AND 200"`).

  For more than one custom attribute statement, place a semicolon between statements (for example, `"Location" = "Bldg. Five";"Service Type" = "eCommerce"`). Multiple "customAttrName:customAttrValue" statements are AND'ed together. Therefore, all the statements must evaluate to true for each node to be included in the Node Group.

- Column 11(K) : Additional Filters "capability" code (Capabilities)

  Optional. List the capabilities assigned to nodes as follows: `capability operator "capability value"[;...]` Note the value must be surrounded by quotes. After importing, these Capabilities specifications appear on the Additional Filters tab of the Node Group form.

  The valid values for operator are as follows:

  `=, !=, like, not like` (If you need other values, use the Node Group form after importing.)

  For more than one capability statement, place a semicolon between statements (for example, `capability = "com.hp.ov.nms.isLANSwitch";capability != "com.hp.ov.nms.isIPv4Router"`). Multiple capability statements are AND'ed together. Therefore, all the statements must evaluate to true for each node to be included in the Node Group.

- Column 12(L) : Security Group Details

  Optional. Specify a list of security-group properties that you want to add to this node-group, separated by semicolon ";". To associate this node-group with a security-group by UUID, enter in the security-group's UUID in the form (xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx), where (x) is a hexadecimal digit (0-9a-fA-F) otherwise it will be assumed that the node-group is to be associated with a security-group by name.

  For example `12345678-1234-1234-123456123456;test_security_group_name`

- Column 13(M) : Tenant Details

  Optional. Specify a list of tenant properties that you want to add to this node-group, separated by semicolon ";". To associate this node-group with a tenant by UUID, enter in the tenant's UUID in the form (xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx), where (x) is a hexadecimal digit (0-9a-fA-F) otherwise it will be assumed that the node-group is to be associated with a tenant by name.

  For example `12345678-1234-1234-123456123456;test_security_group_name`

- Column 14(N) : Node Name

  Optional. Specify a list of nodes by name that you want to add to this node-group, separated by semicolon ";". To associate this node-group with a node by node-name, include the node names in a list of semicolon-separated values.

  For example `test_node_name;node_name_2`

- Column 15(O) : Add to Filter List

  Optional. Sets the Add to Filter List field of the Node Group form.

  `0` (the default setting) means this Node Group is not available as a filter in NNM iSPI Performance reports.

  `1` means this Node Group appears in the Optional Filters selection panel of the NNM iSPI Performance reports.

  Recommendation: Set this value to `1` only for Node Groups that are needed as filters in NNM iSPI Performance reports.

Use of Microsoft Excel

Microsoft Excel is a useful tool to create comma-separated files, but .csv files do not maintain their column width, comments, and other spreadsheet options. HP recommends that you store a `nnmloadnodegroups.ovpl` input file as a native .xls format, then complete a `File:Save As...` command to create a .csv file. You can then add Excel comments to the file, make columns wider, and do not need to escape the comma character. Microsoft Excel also makes it easy to populate the list of child Node Groups. To do this, make column 4(D) contain a calculated value such as the following:

```
=$A1&":0;"&$A2&":0;"&$A3&":0;"&$A4&":0;"&$A5&":0;"&$A6&":0;"&$A7&":0;"
```

This example combines the Node Group Names defined in the first column of the first 7 rows, as child Node Groups of the Node Group Name defined in the first cell of the current row. Using this Excel reference, if you rename the child Node Group in the first column, you do not need to go back and change the reference in the parent Node Group's column 4(D). Note that typing a comma in a line after a leading # in Microsoft Excel generates a non-commented entry when the .xls file is saved as a .csv file (creating a Node Group with a Name starting with the # character).

EXAMPLES

Sample CSV file contents:

```
SNMP,Nodes that support SNMP and that are present in Colorado,,,,,server1.myco.com;server2.myco.com,*.hp.com
```

Note

When entering data in CSV files, do not use the separator characters (":" and ";") for other purposes (for example, in the Names - column 1 - of Node Groups). If you need to use the separator characters, escape them with "\". For example:

- "computer:1" must be entered as "computer\:1"
- "computer;1" must be entered as "computer\;1"
- "computer\:1" must be entered as "computer\\\:1"

To load the Node Groups from a CSV file without overwriting any existing Node Group that matches a Name defined in column 1 of your CSV file:

```
nnmloadnodegroups.ovpl -u system -p myadminpasswd –f /tmp/test.csv
```

To load the Node Groups from a CSV file, and overwrite any existing Node Group that matches a Name defined in column 1 of your CSV file (Caution: this is not a merge, it is a complete replacement of that matching Node Group's configuration):

```
nnmloadnodegroups.ovpl -u system -p myadminpasswd -r true –f /tmp/test.csv
```

AUTHOR

`nnmloadnodegroups.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_BIN%\nnmloadnodegroups.ovpl`

UNIX: `$NNM_BIN/nnmloadnodegroups.ovpl`

SEE ALSO

nnmconfigimport.ovpl(1M), nnmloadinterfacegroups.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmloadseeds.ovpl — load discovery node seed information

SYNOPSIS

`nnmloadseeds.ovpl` -f *`seedFile`* [-t *`tenant`*] | -n *`seeds`* [-t *`tenant`*] [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

DESCRIPTION

`nnmloadseeds.ovpl` allows discovery seeds to be loaded using command line arguments (-n option) or from a text file (-f option). A seed is a device that you want NNM to use as a starting point for the spiral discovery process. Seed values are either IP addresses or host names. When using the -n option the seeds are entered on the command line, separated by white space. Seeds are always added to NNM even if they do not support SNMP

The -f option accepts a file with a single entry specified per line. Each line has the following format:

`IPAddress|HostName`, "*`Optional Tenant Name or UUID`*" # (optional comment to help identify the node, if desired)

Where:

`IPAddress` = the IP address of the node you wish to add.

`HostName` = the host name of the node you wish to add.

Tenants can be optionally specified using either the tenant name or tenant UUID. The tenant specification must be contained within quotation marks. The node discovered from the seed will be assigned to the specified tenant. If no tenant is specified, the node will be assigned to the default tenant.

Comments can be deliminated with a `#` character. Additionally, you can use INCLUDE-FILE *`filename`* to include other seedfiles.

If you specify the -t option, the tenant you specify will be used for all nodes passed in via the -n option, or all nodes in the seed file specified with the -f option. If you use -t and -f and your seed file contains seeds with tenants specified, all seeds with specified tenants will be rejected as invalid seeds.

Note that you should set up the SNMP configuration for the devices being loaded before running this command.

Parameters

The `nnmloadseeds.ovpl` command recognizes the following parameters and options:

`-f` *`seedFile`*

Specify a text file to read the seeds from.

`-n` *`seeds`*

Specify seeds to load directly from the command line, with each seed separated by a space.

`-t` *`tenant name or UUID`*

Specify the tenant to be used for all seeds being loaded.

`-u` <*`username`*>

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p` <*`password`*>

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost` <*`serverName`*>

The server JNDI host; default is `localhost`.

`-jndiPort` <*`port`*>

The server JNDI port; default is `1099`.

EXAMPLES

Load a list of devices as seeds:

```
nnmloadseeds.ovpl -n mimcisco8540 15.2.112.22
```

Load seed for a node using its fully qualified name with username and password:

```
nnmloadseeds.ovpl -u username -p password -n mimcisco8540.superpoller3.mim
```

Load seeds from a file name that is named seeds_to_load.txt:

```
nnmloadseeds.ovpl -f /tmp/seeds_to_load.txt
```

Load seed for a node using its fully qualified name and specific tenant assignment:

```
nnmloadseeds.ovpl -n mimcisco8540.superpoller3.mim -t Customer1
```

Load seeds from a file name that is named seeds_to_load.txt and assign all seeds to a given tenant:

```
nnmloadseeds.ovpl -f /tmp/seeds_to_load.txt -t Customer2
```

AUTHOR

`nnmloadseeds.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `install_dir\bin\nnmloadseeds.ovpl`

UNIX: `$NNM_BIN/nnmloadseeds.ovpl`

SEE ALSO

nnmseeddelete.ovpl(1M), nnmnodedelete.ovpl(1M), nnmnoderediscover.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmmanagementmode.ovpl — Change the NNM management mode of a node or interface.

SYNOPSIS

`nnmmanagementmode.ovpl` [[-node.name *`nodename`* -mode *`mode`*]
| [-t *`object type`* [[-f *`csv_filename`* [-b *`batch size`*]] | [-s *`csv formatted line`*]]]]
[-jdniHost *`hostname`*]
[-jdniPort *`port`*]
[-u *`username`* -p *`password`*]
[-?]

DESCRIPTION

`nnmmanagementmode.ovpl` allows the system administrator to set the management mode of a node or interface in the NNM database.

Parameters

`nnmmanagementmode.ovpl` recognizes the following options. Any unrecognized options are reported by a usage message.

`-?`

Prints the usage statement.

`-node.name` *`nodename`*

The name of a node to change management mode on.

`-mode` *`mode`*

The desired management mode to set. Valid values are "MANAGED", "NOTMANAGED", or "OUTOFSERVICE".

`-t` *`object type`*

Supply the object type to set the management mode on. Valid values are "node" or "interface".

`-f` *`csv_filename`*

Supply the CSV file name (with path, e.g. /opt/tmp/mynodes.csv) from where you want to set the management mode. If the file contains incorrect entries (too many/few columns, incorrect mode) the command will report these but will not execute any of the entries.

`-b` *`batch size`*

If the input comes from a file, the command will process all entries and send the request to the server in chunks specified by this option. The default is 1000 entries at a time.

`-s csv formatted line`

Supply a single CSV formatted line. Eliminates the need to create a file for a simple change

`-jndiHost hostname`

The server jndi host; the default is localhost

`-jndiPort port`

The server jndi port; the default is 1099

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

Syntax of Comma Separated File for node management mode

The CSV file you supply must have the following syntax for setting management mode on nodes.

Empty lines are ignored.

Lines that begin with the # character are ignored.

All columns are mandatory

- Column 1(A) : Node DNS|IP Address

  Specify the DNS name of the node OR the IP address. The search algorithm is as follows:

  - find a node that has a matching management ip address
  - find a node that has a matching ip address in the default domain
  - find a node that has a matching host name
  - find a node that has a matching name

- Column 2(B) : Management mode

  The mode to set the node to. Valid values are "MANAGED", "NOTMANAGED", or "OUTOFSERVICE"

Example lines:

192.168.1.1,OUTOFSERVICE

my.fqdn.com, MANAGED

Syntax of Comma Separated File for interface management mode

The CSV file you supply must have the following syntax for setting management mode on interfaces.

Empty lines are ignored.

Lines that begin with the # character are ignored.

All columns are mandatory

- Column 1(A) : Node DNS|IP Address

  Specify the DNS name of the node OR the IP address. The search algorithm is the same as specified above.

- Column 2(B) : Interface id

  Interface Id Specify the identifier of the interface on the node specified in the previous field. The search algorithm is as follows:

  - ifIndex
  - ifName
  - ifAlias
  - ifDescription

- Column 3(C) : Mode

  The management mode to set the interface to. Valid entries are "INHERITED", "NOTMANAGED", or "OUTOFSERVICE"

Example lines:

192.168.1.1,1,OUTOFSERVICE

my.fqdn.com, myAlias, MANAGED

RETURN VALUE

`nnmmanagementmode.ovpl` exits with the status `0` (zero) if no errors were encountered, 1 otherwise.

AUTHOR

`nnmmanagementmode.ovpl` was developed by Hewlett-Packard Company.

EXTERNAL INFLUENCES

Environmental Variables

Return to Reference Pages Index

Name

nnmnodedelete.ovpl — Remove node(s) and associated data from the NNM topology database

SYNOPSIS

`nnmnodedelete.ovpl` -help | -node <*`hostName`*> | -rm <*`Regional NNMi management server`*> | -file <*`filename`*> [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

DESCRIPTION

`nnmnodedelete.ovpl` removes a node and its associate data like interfaces, ip addresses etc from the system. If this results in empty VLANs and/or subnets they will be removed as well. If incidents point to this node, the Source Node field will be blanked out, but the incidents will not be removed. The node is identified using the hostName field.

The -rm option accepts the name of a Regional NNMi management server. Nodes that are managed by that Regional Manager, will be removed from the local database.

The -file option accepts a file with a single entry specified per line. Each line has the following format:

`HostName #` (optional comment to help identify the node, if desired) Where: *`HostName`* = the host name of the node you wish to add. Comments can be deliminated with a `#` character.

Parameters

`nnmnodedelete.ovpl` supports the following options:

`-node <`*`hostName`*`>`

The hostname of the node to delete.

`-rm <`*`RegionalManagerName`*`>`

The name of the Regional NNMi management server.

`-file <`*`fileName`*`>`

Specify a text file to read the nodes from.

`-u <`*`username`*`>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <`*`password`*`>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost <serverName>`

The server JNDI host; default is `localhost`.

`-jndiPort <port>`

The server JNDI port; default is `1099`.

`-help`

Prints the usage statement.

EXAMPLES

`nnmnodedelete.ovpl -u username -p password -node myNode`

Removes the node `myNode`. (You must provide an NNM username and password.)

`nnmnodedelete.ovpl -u username -p password -rm myRegionalManager`

Removes all nodes associated with `myRegionalManager` (You must provide an NNM username and password.)

`nnmnodedelete.ovpl -u username -p password -file myFile`

Reads the nodes specified in the `myFile` file and attempts to remove them from the database. (You must provide an NNM username and password.)

Diagnostics

`nnmnodedelete.ovpl` returns the following exit codes:

`0`

Operation was successful.

`1`

An error occured; see error message for details.

`2`

Partly successful but some nodes were not deleted; see error messages for details.

AUTHOR

`nnmnodedelete.ovpl` was developed by Hewlett-Packard Company.

FILES

$NNM_BIN/nnmnodedelete.ovpl

NOTES

The deleted node could be rediscovered if it was created due to a `Auto Discovery Rule`. The workaround would be to add the node's ip address to the `Excluded IP Addresses` entry in the `Discovery Configuration` form.

SEE ALSO

nnmseeddelete.ovpl(1M), nnmnoderediscover.ovpl(1M), nnmtopodump.ovpl(1M), nnmresetembdb.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmnodegroup.ovpl — List the names of node groups or print node attributes of nodes belonging to a certain node group.

SYNOPSIS

`nnmnodegroup.ovpl`
{ -?
| -list
| -printNodes <*group name*> [-hostName | -shortName | -uuid | -ip]}
[-jdniHost *hostname*]
[-jdniPort *port*]
[-u *username* -p *password*]

DESCRIPTION

`nnmnodegroup.ovpl` will list all node groups in the database or print node attributes belonging to a certain node group. When printing the node attributes, the group name is a required argument; if no other arguments are given, `the hostName, shortName, uuid and management IP address` attributes of each node are dumped; one comma separated line per node.

Parameters

`nnmnodegroup.ovpl` supports the following options:

`-list`

Print the name of the node groups in the database.

`-printNodes <node group name> [-hostName | -shortName | -uuid | -ip]`

Print attributes of nodes belonging to the specified `node group`. If no additional parameters are provided, the `hostName, shortName, uuid and management ip address` attributes of each node will be printed; one comma separated line per node.

Optionally one of the following parameters can be specified, only the specified attribute will be printed:

`-hostName`

Prints the hostName of each node belonging to the node group.

`-shortName`

Prints the short name of each node belonging to the node group.

`-uuid`

Prints the UUID of each node belonging to the node group.

`-ip`

Prints the management IP address of each node belonging to the node group. If this can not be determined, the host name will be printed instead.

`-?`

Prints the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost`

The server JNDI host; default is `localhost`.

`-jndiPort`

The server JNDI port; default is `1099`.

EXAMPLES

To print the hostName, shortName, uuid and management ip address of all nodes that belong to the node group `Routers`:

```
nnmnodegroup.ovpl -printNodes Routers
```

To only print the host name of all nodes that belong to the node group `Non-SNMP Devices`:

```
nnmnodegroup.ovpl -printNodes "Non-SNMP Devices" -hostName
```

To list the node group names in the database:

```
nnmnodegroup.ovpl -list
```

RETURN VALUE

`nnmnodegroup.ovpl` exits with the status `0` (zero) if no errors were encountered, 1 otherwise.

AUTHOR

`nnmnodegroup.ovpl` was developed by Hewlett-Packard Company.

FILES

`$NNM_BIN/nnmnodegroup.ovpl`

Return to Reference Pages Index

Name

nnmnoderediscover.ovpl — Discover node details after the node has changed

SYNOPSIS

`nnmnoderediscover.ovpl` -help | -node <*`hostName`*> [-tenant <*`name`*>] [-fullsync] | -rm <*`Regional NNMi management server`*> [-fullsync] | -file <*`filename`*> [-tenant <*`name`*>] [-fullsync] | -all [-fullsync] [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

DESCRIPTION

`nnmnoderediscover.ovpl` places node(s) into the NNMi discovery queue. The amount of time before the node starts discovery is dependent on how long NNMi takes to get to the node in the queue.

If the node is already in the discovery queue, it is not added again.

Note that when using the NNMi Advanced Global Network Management feature, nodes that are managed by a Regional Manager only go through discovery on the Regional Manager (NNMi management station) and are not rediscovered by the Global Manager.

The -rm option is for the NNMi Advanced Global Network Management feature. It is used when issuing the `nnmnoderediscover.ovpl` script on a Global Manager. NNMi requests that the Regional Manager (NNMi management server) send the most recent discovery results to the Global Manager.

The -file option accepts a file with a single entry specified per line. Each line contains a short name, or a fully-qualified DNS domain name, or an IP address. Each line has the following format: HostName # (optional comment to help identify the node) Where: HostName = the short name or DNS name or IP address of the node you want to add.

The -all option causes all nodes managed by the local NNMi management server to be rediscovered. Note that when you use the NNMi Advanced Global Network Management feature and issue the `nnmnoderediscover.ovpl` script on a Global Manager, the most recent discovery results are sent from the Regional Manager to the Global Manager.

The -tenant option identifies nodes with non-unique names or IP addresses such as might can the case with overlapping address domains. The name passed with the argument is the quoted tenant name.

The -fullsync option causes resynchronization to the node's states and status following the rediscovery of the node. When run from a Global Manager, the Global Manager's nodes are updated from the Regional Manager's nodes. When run from a Regional Manager, the command does a resynchronization for the regional node(s) and also does a resynchronization on the Global Manager for those nodes that belong to the given Regional Manager. This is an optional flag, which does not affect the algorithm for choosing which nodes are of rediscovered.

Parameters

`nnmnoderediscover.ovpl` supports the following options:

`-node <hostName>`

The hostname of the node to rediscover.

`-rm <RegionalManagerName>`

When using the NNMi Advanced Global Network Management feature and issuing the `nnmnoderediscover.ovpl` script on a Global Manager, the Name attribute value identifies a Regional NNMi management server.

`-file <fileName>`

Specifies a text file to read the nodes from.

`-all`

Specifies all nodes are to be rediscovered.

`-tenant <name>`

Optional parameter that when paired with a node name or IP will identify the node in domains where the name or IP can be non-unique.

`-fullsync`

Optional parameter that directs NNMi to resynchronize a node's states and status following the rediscovery of each node.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost <serverName>`

The server JNDI host; default is `localhost`.

`-jndiPort <port>`

The server JNDI port; default is `1099`.

`-help`

Displays the usage statement.

EXAMPLES

`nnmnoderediscover.ovpl -u username -p password -node myNode -tenant myTenant`

Rediscovers the node `myNode` that is a member of the tenant `myTenant`. You must provide a valid administrator *`username`* and *`password`*.

```
nnmnoderediscover.ovpl -u username -p password -rm myRegionalManager
```

Rediscovers all nodes associated with `myRegionalManager`. You must supply a valid NNMi administrator *`username`* and *`password`*.

```
nnmnoderediscover.ovpl -u username -p password -file myFile -fullsync
```

Reads the nodes specified in the `myFile` file and places them into the NNMi discovery queue. Following the rediscovery of each node, the state and status for the node will be resynchronized. You must provide a valid adminstrator *`username`* and *`password`*.

Diagnostics

`nnmnoderediscover.ovpl` returns the following exit codes:

`0`

Operation was successful.

`1`

An error occurred; see error message for details.

AUTHOR

`nnmnoderediscover.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnmdiscocfg.ovpl(1M), nnmloadseeds.ovpl(1M), nnmseeddelete.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmofficialfqdn.ovpl — Display the official Fully Qualified Domain Name (FQDN) of the NNMi management server.

SYNOPSIS

```
nnmofficialfqdn.ovpl
```

DESCRIPTION

Use `nnmofficialfqdn.ovpl` to display the official Fully Qualified Domain Name (FQDN). The official FQDN must be correct and consistent across systems to enable Single Sign On with NNM iSPIs. NNMi sets the official FQDN during installation. After installation, you can change it using the nnmsetofficialfqdn.ovpl command.

Options

`nnmofficialfqdn.ovpl` by default displays verbose output with a ping test. `nnmofficialfqdn.ovpl` recognizes the following options:

`-d`

Display the domain name of the official FQDN. If the FQDN is only a short hostname or an IP address and if the terse option (-t) is used, NNMi does not display a value; otherwise, NNMi displays a message informing you that the domain name cannot be found.

`-t`

Terse mode. Displays the FQDN or domain name value only. When the FQDN or domain name is not found, no warning or informative text is shown. This option also skips the ping test.

`-m`

Query for the default FQDN and display the value. NNMi displays whichever value it finds first using the following order:

- FQDN
- Short hostname
- IP address

If none of the above are found, then 'localhost' is displayed.

`-h`

Display the help menu listing all the options.

EXAMPLES

Running the command without any options displays the official FQDN and performs a ping test:

```
# nnmofficialfqdn.ovpl
      FQDN: hostname.somedomain
      Ping test: pinging hostname.somedomain please wait...
      Ping OK
```

Running the command with the -t option displays the official FQDN as follows:

```
# nnmofficialfqdn.ovpl -t
      hostname.somedomain
```

Running the command using the -d option displays the domain name as follows:

```
# nnmofficialfqdn.ovpl -d
       Domain: somedomain
```

Running the command using the -d and -t options displays only the domain name:

```
# nnmofficialfqdn.ovpl -dt
      somedomain
```

AUTHOR

`nnmofficialfqdn.ovpl` was developed by Hewlett-Packard Company.

FILES

`nnmofficialfqdn.ovpl` resides in `$NNM_BIN directory`.

SEE ALSO

nnmsetofficialfqdn.ovpl(1M), nnmsso.ovpl(1M).

Return to Reference Pages Index

Name

nnmooflow.ovpl — Manage available HP OO flow definitions used by NNMi iSPI NET.

SYNOPSIS

`nnmooflow.ovpl` [-h] [-l] [-i *`xmlfilename`*] [-d *`uuid`*]

DESCRIPTION

Manage user defined HP OO flow definitions. This command can be used to list current HP OO flow definitions available to NNMi, import a new HP OO flow definition to NNMi, or delete an existing HP OO flow definition from NNMi.

HP OO flows are authored using HP Operations Orchestration Studio. Flows in HP OO can be invoked from NNMi during incident processing. A number of arguments can be passed to HP OO flows. See below for the full list of arguments which can be passed.

This list option will show all available HP OO flow definitions including those shipped with the iSPI NET diagnostics server and those previously imported. Note that only user defined HP OO flow definitions can be deleted.

An HP OO flow definition is imported into NNMi by creating an xmlfile using the format described below and importing the definition using the -i option. A single xml file can contain a single flow definition. If multiple flow definitions need to be imported to NNMi, create a separate xml file for each.

The following is a sample import xmlfile

```
<definition xmlns="http://openview.hp.com/xmlns/nnmi/diagnostics/1">
    <uuid>a2dbd722-b0c3-43ad-b435-4eae9adf37b1</uuid>
    <name>Test Flow</name>
    <description>A test flow for import</description>

    <arguments>
        <argument>
            <type>HOST</type>
            <name>hostname</name>
            <required>true</required>
        </argument>

        <argument>
            <type>SHELL_USERNAME</type>
            <name>user</name>
            <required>false</required>
        </argument>

        <argument>
            <type>SHELL_PASSWORD</type>
            <name>password</name>
            <required>true</required>
        </argument>

    </arguments>

    <filters>
        <filter>
            <vendor>com.hp.ov.nms.devices.cisco</vendor>
            <category>com.hp.ov.nms.devices.router</category>
```

```
<family>com.hp.ov.nms.devices.cisco3600seriesmultiservicebranchofficerouter</family>
        </filter>
    </filters>
 </definition>
```

The XML document requires the following structure:

```
<definition>
    <uuid></uuid>
    <name></name>
    <description></description>
    <arguments>
        <argument></argument>
    </arguments>
    <filters>
        <filter></filter>
    </filters>
</definition>
```

<definition> - XML document element.

<uuid>nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn</uuid>- contains the uuid of the HP OO flow as provided by HP OO Studio. The uuid is available on the properties tab for the flow displayed in HP OO Studio.

<name>string</name> - contains the name of the flow to be displayed in NNMi. Name can be a maximum of 128 characters. More than 128 characters will cause a parse exception when reading the xml file.

<description>string</description> - contains the description of the flow to be displayed in NNMi. Description can be a maximum size of 2048 characters. More than 2048 characters will cause a parse exception when reading the xml file.

<arguments> - contains a list of 1 or more <argument> elements to pass to the flow. Include as many <argument> children as necessary. Each flow defined in HP OO Studio can accept named inputs. The <arguments> list allows HP NNMi to pass NNMi defined variables to these inputs. The <type> element defines the variable NNMi will pass to the input identified in the <name> element. SHELL_USERNAME and SHELL_PASSWORD are defined using NNMi's creditional configuration in NNMi's Communication configuration workspace. The variables passed to the HP OO flow are determined based on the incident's related device, interface, or port. If a argument is specified as required but NNMi can't determine an appropriate value based on the incident, the HP OO flow will not be invoked.

An <argument> element contains a <type>, a <name>, and <required> elements.

The <type> element can be any of the following: SHELL_USERNAME, SHELL_PASSWORD, HOST, IFACE_IFNAME, IFACE_IFTYPE, IFACE_IFALIAS, IFACE_IFINDEX, PORT_NAME, PORT_INDEX.

The <name> element represents the name of the input as defined in the HP OO Flow. The variable will be passed to the HP OO Flow using this input name.

The <required> element may be true or false and indicates if the argument must be supplied to HP OO.

Multiple <filter> elements can be specified. <filter> elements define the types of devices where a flow can be excuted. <filter> elements are constructed using <vendor>, <category>, and <family> child elements.

Filters specify the vendor, category, and family of the device or devices the flow supports. As such flows can be defined for generic use across a vendor's product line or defined to more narrowly execute on a category or family of devices from a vendor. Use the "unique key" as indentified in sub forms for Device Profile. You can find the "unique key" by opening the form corresponding to a specific Device vendor, Device Category, or Device Family using the Device Profile workspace.

A filter can specify just vendor, category, or family or all three to indicate a very specific device profile. <filters> can be empty, if the flow can operate on any device.

Once a flow definition has been imported to NNMi, it is available to be used in NNMi Incident Configuration. See NNMi online help for instructions on configuring incident flows.

Parameters

`nnmooflow.ovpl` supports the following parameters:

`-h`

Displays command summary

`-l`

Displays a list of all HP OO flow definitions available for use in incident configuration.

`-d` *`uuid`*

Removes the flow definition identified by the uuid. HP defined flow definitions can not be removed. User defined flow definitions which are referenced in incident configuration also may not be removed.

`-i` *`xmlfilename`*

Imports the flow definition defined by the xml file. A flow can be repeatly imported to NNMi. Only the last flow definition for a uuid imported by this command will be retained in the configuration.

EXAMPLES

To list currently available HP OO Flow Definitions:

```
nnmooflow.ovpl -l
```

To load a flow definition

```
nnmooflow.ovpl -i flow.xml
```

To delete a flow definition

```
nnmooflow.ovpl -d 01d616b0-e852-4235-b447-48185bd31444
```

AUTHOR

`nnmooflow.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmopcexport.ovpl — export NNMi management event and SNMP trap configurations to HP OM policies (SNMP template) file.

SYNOPSIS

`nnmopcexport.ovpl` -help

`nnmopcexport.ovpl` [-u <*username*> -p <*password*>] [-jndiHost <*hostname*> -jndiPort <*port*>] -template <*template_name*> -application <*application_name*> [-agtmsi_copy | -agtmsi_divert] [-svmsi_copy | -svmsi_divert] [-msgtype] [-author <*author code*>] [-oid <*snmp oid prefix*>] [-omi_hi] -file <*output_file name*> [-force]

`nnmopcexport.ovpl` [-u <*username*> -p <*password*>] [-jndiHost <*hostname*> -jndiPort <*port*>] -template <*template_name*> -application <*application_name*> [-agtmsi_copy | -agtmsi_divert] [-svmsi_copy | -svmsi_divert] [-msgtype] [-author <*author code*>] [-oid <*snmp oid prefix*>] [-omi_hi] -omi_policy

DESCRIPTION

`nnmopcexport.ovpl` reads the NNMi management event and SNMP trap configurations and exports the configurations as an HP OM policies file. This tool provides similar functionality to the HP OM ovtrap2opc utility that converted legacy NNM trapd.conf files into HP OM policies.

Parameters

`nnmopcexport.ovpl` supports the following parameters:

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-agtmsi_copy`

Copy matching messages to the OVO Agent Message Stream Interface.

`-agtmsi_divert`

Divert matching messages to the OVO Agent Message Stream Interface.

`-application <application_name>`

Set the policy application name.

`-author <author_code>`

Export events matching the author code.

`-file <output_filename>`

Save output to file with the specified name.

`-force`

Overwrite an existing policy file specified by -file `<output_file_name>`.

`-jndiHost <hostname>`

The server jndi host; default is localhost.

`-jndiPort <port>`

The server jndi port; default is 1099.

`-msgtype`

Set the message type to the name of the condition. The value of the resulting MSGTYPE field will be truncated to 32 characters for OM import compatibility.

`-oid <snmp_oid_prefix>`

Export events with snmp trap OIDs matching the provided OID.

`-omi_hi`

Include HP OMi health indicators for applicable NNMi management events.

`-omi_policy`

Export in OMi policy format. Using the `-omi_policy` option is the recommended usage. Only generate policies without this option if you need to maintain backwards compatibility with previous OM integrations. A header file and a data file are created in the form <UUID>_header.xml and <UUID>_data where UUID is a Universally Unique Identifier. The -file option is ignored. Using the `-omi_policy` option generates policies using the `RelatedCiInfo` OMi custom attribute. If you do not use the `-omi_policy` option, the policy uses the older `OPR_CI_INFO` OM custom attribute.

`-svmsi_copy`

Copy matching messages to the OVO Server Message Stream Interface.

`-svmsi_divert`

Divert matching messages to the OVO Server Message Stream Interface.

`-template template name`

Set the <template_name>.

`-help`

Show help.

EXAMPLES

Export an OM policy file containing all NNMi management events and SNMP trap definitions:

```
nnmopcexport.ovpl -u user -p pass -template "SNMP Traps" -application "NNMi" -file
policies.dat
```

Export an OMi policy file containing all NNMi management events and SNMP trap definitions and OMi health indicators:

```
nnmopcexport.ovpl -u user -p pass -template "SNMP Traps" -application "NNMi" -omi_hi -
omi_policy
```

AUTHOR

`nnmopcexport.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmperfspisync.ovpl — Force the PerfSpiAdaptor to dump NNMi's node and interface groups so Performance SPIs are consistent with NNMi.

SYNOPSIS

`nnmperfspisync.ovpl`

DESCRIPTION

`nnmperfspisync.ovpl` is a Perl script that enables you to force the PerfSpiAdaptor to dump node and interface groups within 5 minutes.

NOTE: It can take up to 5 minutes for the group topology to be written to the file system.

Parameters

The `nnmperfspisync.ovpl` script does not support any parameters.

NOTES

The `nnmperfspisync.ovpl` script will not overwrite an existing dump request if one exists, over-writing will not speed up the 5 minute timer for the dump request.

EXAMPLES

`nnmperfspisync.ovpl`

Starts a 5 minute timer for a full node and interface group dump, if it does not already exist.

AUTHOR

`nnmperfspisync.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmresetembdb.ovpl — drop (delete) and recreate the embedded database, if NNMi is configured to run with the embedded database

SYNOPSIS

`nnmresetembdb.ovpl` [-?|-h|-help] [-silent] [-nostart]

DESCRIPTION

Use `nnmresetembdb.ovpl` to drop (delete) and recreate the NNMi embedded database. This command is useful only if you installed NNMi with the embedded database option. You should use this command only if your database is corrupt, and you are willing to lose all of your data; or, if you simply want to reset your database to the state it was in after initial installation.

If NNMi is running when this command is executed, the command first stops NNMi (using `ovstop`), then starts the `nmsdbmgr` process to work with the database (using `ovstart`). Unless the `-nostart` option is present, NNMi restarts (using `ovstart`) upon successful completion of the reset process.

When the database reset has completed, the embedded database has no tables or data in it. The tables are recreated when you restart NNMi using the `ovstart` command, or when the `nnmresetembdb.ovpl` command automatically starts NNMi. You must be logged in as `root` on UNIX systems, or as `administrator` on Windows systems to run this command.

Parameters

The `nnmresetembdb.ovpl` command recognizes the following parameters:

`-silent`

> The `nnmresetembdb.ovpl` command does not display the command results when you use the `-silent` option.

`-nostart`

> The `nnmresetembdb.ovpl` command does not start NNMi after the database reset when you use the `-nostart` option.

`-?|-h|-help`

> Display command usage.

EXAMPLES

You can use this script to reset your database if you have a bad discovery or a corrupted database; or if you

want to reset the database (including configuration items stored in the database) to the state the database was in after initial installation.

You'll see these messages:

```
# nnmresetembdb.ovpl -nostart

WARNING: Running this tool will stop NNM, drop and recreate the database,
and restart NNM. Please make sure no major activity is occurring at
this time.

Is it OK to stop NNM (ovstop)? [n] y
Thank you!

WARNING: This will delete all configuration and discovered data. You cannot
recover from a reset unless you have taken a backup.

Are you sure you wish to reset your database? [n] y

Attempting to reset the embedded database...
Starting nmsdbmgr process for database reset...
Successfully started nmsdbmgr process for database reset.
Successfully reset private database.
Successfully reset NNM embedded database.

#
```

AUTHOR

`nnmresetembdb.ovpl` was developed by Hewlett-Packard Company.

FILES

`nnmresetembdb.ovpl` resides in `$NNM_BIN` (UNIX) or *`install_dir`*`\bin` (Windows) directories.

SEE ALSO

nmsdbmgr(1M), nnmnodedelete.ovpl(1M), ovstart(1M), ovstop(1M), ovstatus(1).

Return to Reference Pages Index

Name

nnmrestore.ovpl — restore a backup created by the `nnmbackup.ovpl` script

SYNOPSIS

`nnmrestore.ovpl` [-?|-h|-help] [-force] [-lic] [-partial] -source <*`directory`*>

DESCRIPTION

`nnmrestore.ovpl` is the main restore script for NNMi. The command uses a previous NNMi backup created using the `nnmbackup.ovpl` script to restore NNMi to the state stored in the backup files. The scope of the restore is determined by the content of the backup and by the command-line arguments you provide. The `nnmrestore.ovpl` script restores only the data contained in the backup files.

If you plan to use the `nnmbackup.ovpl` script to create an NNMi backup, then use the `nnmrestore.ovpl` script to place database records on a second NNMi management server, both NNMi management servers must have the same type of operating system and NNMi version and patch level. Placing the backup data from one NNMi management server onto a second NNMi management server means that both servers have the same database UUID. After you restore NNMi on the second NNMi management server, uninstall NNMi from the original NNMi management server.

The `nnmrestore.ovpl` script detects the following:

- Was the backup performed on the target system? If the backup was taken locally, the `-lic` parameter permits the restoration of licensing information.
- Is the target system running an Oracle or Postgres database? If you require a full restore of an Oracle database, for example, running the `nnmrestore.ovpl` script without the `-partial` parameter, the `nnmrestore.ovpl` script prompts you for information to determine if the target system is using the same Oracle database instance as the backup, or if an Oracle backup and restore operation has already taken place on the target system. You must answer these questions correctly or the script will copy or merge the SSL certificates incorrectly and the database will become unaccessible from the target installation.

The source directory contains either all of the files required for the restore options you have selected, or a single tar file. If the source is a tar file, the `nnmrestore.ovpl` script extracts the tar file to a temporary folder in the current working directory. The `nnmrestore.ovpl` script removes the temporary folder after completing the restore.

NNMi must be stopped to complete restore operations. If you use the `–force` option, the `nnmrestore.ovpl` script stops NNMi. If the files present in the source folder indicate that the original backup was an `online` backup, the restore process starts the `nmsdbmgr` process to make sure the embedded database is available. To restore the files from an online backup, you must use the `-force` option.

You must be logged in as as `administrator` on Windows systems or `root` on UNIX systems to run the `nnmrestore.ovpl` script.

Caution

The database restore performed by the `nnmrestore.ovpl` script applies only to the embedded database. If you chose a different database at installation, the table data is not restored using the `nnmrestore.ovpl` script. File-system restores work regardless of database type. For details about what you need to restore if you select a different database at installation, refer to the *NNMi Deployment Reference* (available at http://h20230.www2.hp.com/selfsolve/manuals).

Parameters

`nnmrestore.ovpl` supports the following options:

`-force`

If you provide this option, the `nnmrestore.ovpl` script stops NNMi before running the restore procedure. Also, the script starts the `nmsdbmgr` process if it must restore the type of backup on which the restore is based. This restore from an online backup to use this option.

`-lic`

If you provide this option, the `nnmrestore.ovpl` script restores licensing information. Note that the script only restores licensing information if it successfully validates that the backup operation was performed on the same system.

`-partial`

Without this option, the `nnmrestore.ovpl` script restores the database and corresponding SSL certificates. In the case of a system-to-system restore, the `nnmrestore.ovpl`script merges the `nnm.keystore` and `nnm.truststore` files with the target system's `nnm.keystore` and `nnm.truststore` files. The merge routine merges all certificate aliases from the backed up stores that do not already exist on the target system. There is one exception, if the `FQDN.selfsigned` key alias exists in both stores, the target system's key alias is removed and replaced with the one stored in the backup. Before the certificates are merged or restored, a backup is performed, stored in the same directory, and labeled `*.original`.

When you use the `-partial` option, the `nnmrestore.ovpl` script will not restore the database and corresponding SSL certificates. The `-partial` option is only useful for restoring configuration files.

`-source<`*`directory`*`>`

Specifies the input directory containing the restore files. If the source you provide is a tar file, the `nnmrestore.ovpl` script extracts the tar file to a temporary folder in the current working directory. The script removes the temporary folder after completing the restore.

`-?|-h|-help`

Display command usage.

EXAMPLES

To restore a previous backup:

```
#./nnmrestore.ovpl –source /tmp/bak/config
```

To restore with the `-force` option:

```
#./nnmrestore.ovpl –force –source /tmp/bak/all
```

To restore everything except the database, SSL certificates, and licenses, use the `-partial` option:

```
#./nnmrestore.ovpl –partial –source /tmp/bak/all
```

To restore everything, including licensing information on a local system, use the `-lic` option:

```
#./nnmrestore.ovpl –lic –source /tmp/bak/all
```

AUTHOR

`nnmrestore.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnmbackup.ovpl(1M).

Return to Reference Pages Index

Name

nnmrestoreembdb.ovpl — restore a full backup of the NNMi embedded database

SYNOPSIS

`nnmrestoreembdb.ovpl [-?|-h|-help] [-force] -source <backup file>`

DESCRIPTION

Use the `nnmrestoreembdb.ovpl` script to restore a full backup of the NNMi embedded database. Create the backup file required for the restore using the `nnmbackupembdb.ovpl` script. Because query planning statistics are not backed up, these statistics are gathered as part of the restore procedure to ensure good database performance after a restore.

If you plan to use the `nnmbackupembdb.ovpl` script to create a backup of the NNMi embedded database, then use the `nnmrestoreembdb.ovpl` script to place embedded database records on a second NNMi management server, both NNMi management servers must have the same type of operating system and NNMi version and patch level.

Before running the `nnmrestoreembdb.ovpl` script, make sure the embedded database is empty. You can do this by running the `nnmresetembdb.ovpl` script. If you do not clear the embedded database, the restore procedure runs the `nnmresetembdb.ovpl` script for you and the restore fails if the `nnmresetembdb.ovpl` script fails.

Do *not* run this script while NNMi is running unless you specify the `-force` option. Only the `nmsdbmgr` process can (and must) be running when you run the `nnmrestoreembdb.ovpl` script. You must be logged in as as administrator on Windows systems or root on UNIX systems, to run this script.

Parameters

`-source <backup file>`

> The file to use when restoring the backup. This file must be a file that was created using the `nnmbackupembdb.ovpl` script.

`-force`

> If you provide this option, the script stops NNMi if it is currently running, then starts the `nmsdbmgr` process.

`-?|-h|-help`

> Display script usage.

EXAMPLES

Use this script to run a full database recovery when a restore from backup becomes necessary.

You will see these messages:

```
# nnmrestoreembdb.ovpl -source /backups/nnmDb.dump

WARNING: Running this command while NNMi is running will cause sporadic
         failures in active sessions. Please ensure that only the nmsdbmgr
         process is running when executing this command (ovstart nmsdbmgr).
Are you sure you want to run a full database restore now? [n] y

Ok, performing full embedded database restore...
Statistics analysis completed successfully.

NNMi embedded database successfully restored from /backups/nnmDb.dump.
#
```

AUTHOR

`nnmrestoreembdb.ovpl` was developed by Hewlett-Packard Company.

FILES

`nnmrestoreembdb.ovpl` resides in the following directories:

- *Windows*: `%NNM_BIN%`
- *UNIX*: `$NNM_BIN`

SEE ALSO

ovstart(1M), ovstop(1M), ovstatus(1M), nmsdbmgr(1M), nnmbackupembdb.ovpl(1M).

Return to Reference Pages Index

Name

nnmsecurity.ovpl — NNMi Security Management

SYNOPSIS

`nnmsecurity.ovpl` -help

`nnmsecurity.ovpl` -assignNodeToSecurityGroup ((-node <*name or hostname or management address or uuid*> -securityGroup <*name or uuid*>) | -file <*name*>) | -assignNodeToTenant ((-node <*name or hostname or management address or uuid*> -tenant <*name or uuid*>) | -file <*name*>) | -assignSecurityGroupToTenant (-tenant <*name or uuid*> -securityGroup <*name or uuid*>) | -assignUserGroupToSecurityGroup ((-userGroup <*name*> -securityGroup <*name or uuid*> -role <*role*>) | -file <*name*>) | -assignUserToGroup ((-user <*name*> -userGroup <*name*>) | -file <*name*>) [-u <*username*> -p <*password*>] [-jndiHost <*hostname*> Default: localhost] [-jndiPort <*port*> Default: 1099]

`nnmsecurity.ovpl` -createSecurityGroup ((<*name*> [-securityGroupUuid <*uuid*>] [-description <*description*>]) | -file <*name*>) | -createTenant (<*name*> [-tenantUuid <*uuid*>] [-securityGroupUuid <*uuid*>] [-description <*description*>]) | -createUserAccount ((<*username*> -role <*role*> [-password <*password*>] [-directoryServiceAccount <*true|false*>]) | -file <*name*>) | -createUserGroup ((<*name*> [-displayName <*user friendly group name*>] [-description <*description*>] [-directoryServiceName <*dn*>]) | -file <*name*>) [-u <*username*> -p <*password*>] [-jndiHost <*hostname*> Default: localhost] [-jndiPort <*port*> Default: 1099]

`nnmsecurity.ovpl` -deleteSecurityGroup (<*groupName or uuid*> | -file <*name*>) | -deleteUserAccount (<*name*> | -file <*name*>) | -deleteUserGroup <*name*> [-u <*username*> -p <*password*>] [-jndiHost <*hostname*> Default: localhost] [-jndiPort <*port*> Default: 1099]

`nnmsecurity.ovpl` -displayConfigReport [<*report*>[, <*report*>]] [-u <*username*> -p <*password*>] [-jndiHost <*hostname*> Default: localhost] [-jndiPort <*port*> Default: 1099]

`nnmsecurity.ovpl` -listNode <*nodeName*> | -listNodesInSecurityGroup <*groupName or uuid*> | -listSecurityGroupForTenant <*uuid*> | -listSecurityGroups | -listTenants | -listUserGroupMembers <*groupName*> | -listUserGroups <*user*> | -listUserGroupsForSecurityGroup <*groupName or uuid*> [-u <*username*> -p <*password*>] [-jndiHost <*hostname*> Default: localhost] [-jndiPort <*port*> Default: 1099]

`nnmsecurity.ovpl` -removeUserFromGroup ((-user <*name*> -userGroup <*name*>) | -file <*name*>) | -deleteUserGroup (<*name*> | -file <*name*>) | -removeUserGroupFromSecurityGroup ((-userGroup <*groupName*> -securityGroup <*groupName or uuid*> [-role <*role*>]) | -file <*file*>) | -updateUserGroup ((<*name*> [-displayName <*user friendly group name*>] [-description <*description*>] [-directoryServiceName <*dn*>]) | -file <*name*>) [-u <*username*> -p <*password*>] [-jndiHost <*hostname*> Default: localhost] [-jndiPort <*port*> Default: 1099]

DESCRIPTION

If you frequently run NNMi command line tools, create an `nnm.properties` file containing your username and password. Doing so permits you to run many NNMi command line tools and scripts without entering a username and password. Place the `nnm.properties` file in a `.nnm` subdirectory within your home directory.

For example, you might place the `nnm.properties` file you create in the *`drive`*`:\Documents and Settings\`*`username`*`\.nnm\` (Windows) or `~/.nnm` (UNIX) directory.

`nnmsecurity.ovpl` is used to manage NNMi security configuration. It provides commands to create, update, and remove security objects such as user accounts, user groups, and security groups as well as to configure the relationships among these objects. This command replaces the deprecated nnmprincipalconfig.ovpl command.

Parameters

`nnmsecurity.ovpl` supports the following commands:

`-assignNodeToSecurityGroup (-node <`*`name or hostname or management address or uuid`*`> -securityGroup <`*`name or uuid`*`>) | -file <`*`name`*`>`

Assigns nodes to security groups using either command line arguments or an input file.

`-node`

Identifies a node by name, hostname, management address, or UUID.

`-securityGroup`

Identifies a security group by name or UUID.

`-file`

Path to a CSV-formatted file containing lists of node to security group assignment with the format: `securitygroup, node`

`-help`

Prints the usage statement.

`-assignNodeToTenant (-node <`*`name or hostname or management address or uuid`*`> -tenant <`*`name or uuid`*`>) | -file <`*`name`*`>`

Assigns a node to a tenant using either command line arguments or an input file. The node-to-tenant assignment must be done on an NNMi management server that directly manages both objects. Global node-to-tenant assignments are unsupported.

`-node`

Identifies a node by name, hostname, management address, or UUID.

`-tenant`

Identifies a tenant by name or UUID.

`-file`

Path to a CSV-formatted file containing lists of node to tenant assignments with the format: `node,tenant`

`-assignSecurityGroupToTenant -tenant <`*`name or uuid`*`> -securityGroup <`*`name or uuid`*`>`

Changes the default security group for a tenant. The default security group for a tenant is used to specify which security group to use when new nodes are seeded for the tenant. Changing this value

does not affect existing nodes.

`-tenant`

The name or UUID of the tenant to modify.

`-securityGroup`

The name or UUID of the security group to set as the default for the tenant.

`-assignUserGroupToSecurityGroup (-userGroup <name> -securityGroup <name or uuid> -role <role>) | -file <name>`

Assigns user groups to security groups. User groups are assigned to security groups to give the users in the group access to the nodes in the security group. Each assignment includes a role as part of the assignment which controls which actions are available to the users on the nodes.

`-userGroup`

Identifies the user group to assign by name.

`-securityGroup`

Identifies by name or UUID the security group to receive the user group.

`-role`

Identifies the role to use in the assignment by key. Available roles are: `admin, level2, level1, guest`

`-file`

Path to a CSV-formatted file containing lists of assignments with the format: `userGroup, securityGroup, role`

`-assignUserToGroup (-user <name> -userGroup <name>) | -file <name>`

Assigns users to user groups. Users are assigned to groups which are then given access to objects. A user can be assigned to multiple groups and has access to all objects from all of their groups. The default groups of admin, client, level2, level1 and guest also give the users assigned to them the matching role of the same name on NNMi itself.

`-user`

Identifies the user to assign by name.

`-userGroup`

Identifies the user group to assign by name.

`-file`

Path to a CSV-formatted file containing lists of assignments with the format: `user, userGroup`

`-createSecurityGroup (<name> [-securityGroupUuid <uuid>] [-description <description>]) | -file <name>`

Creates a new security group. Security groups group similar topology objects to simplify the security configuration. Each security group consists of a name, UUID, and description.

`-securityGroupUuid`

Optional UUID for the new security group. If this parameter is not supplied, NNMi generates the value.

`-description`

Optional description for the new security group.

`-file`

Path to a CSV-formatted file containing lists of security groups with the format: `name, uuid, description`

`-createTenant <name> [-tenantUuid <uuid>] [-securityGroupUuid <uuid>] [-description <description>]`

Creates a new tenant along with a matching security group of the same name.

`-tenantUuid`

Optional UUID for the new tenant. If this parameter is not supplied, NNMi generates the value.

`-securityGroupUuid`

Optional UUID for the new security group. If this parameter is not supplied, NNMi generates the value.

`-description`

Optional description for the new tenant.

`-createUserAccount (<username> -role <role> [-password <password>] [-directoryServiceAccount <true|false>]) | -file <name>`

Creates a new user account.

`-role`

Internal accounts require that a role be specified. NNMi automatically assigns the new user to the matching user group. External accounts do not require a role because the directory service might supply the roles.

`-password`

The password for the new user. Only used for internal accounts.

`-directoryServiceAccount`

Specifies whether an external directory service manages this user account. Use `false` for an account that is stored internally in the NNMi database. Use `true` for an external account that is stored in a directory service. The default value is `false`.

`-file`

Path to a CSV-formatted file containing lists of user accounts with the format: `username, password, role, directoryServiceAccount`

`-createUserGroup (<name> [-displayName <user friendly group name>] [-description <description>] [-directoryServiceName <dn>]) | -file <name>`

Creates a new user group.

`-displayName`

Optional friendly name for the user group.

`-description`

Optional description of the new group.

`-directoryServiceName`

Optional for directory service users. Use this option to pair a directory service distinguished name with this user group.

`-file`

Path to a CSV-formatted file containing lists of user groups with the format: `name, displayName, description, directoryServiceName`

`-deleteSecurityGroup <groupName or uuid> | -file <name>`

Removes a security group by name or UUID. The security group must not have any nodes or tenants assigned to it.

`-file`

Path to a CSV-formatted file containing lists of security groups with the format: `name, uuid, description`. This format is the same as for createSecurityGroup however only the name (or UUID if present) is used.

`-deleteUserAccount <name> | -file <name>`

Removes a user account by name.

`-file`

Path to a CSV-formatted file containing lists of user accounts with the format: `username, role, password, directoryServiceAccount`. This format is the same as for createUserAccount however only the username is used to match the accounts to remove.

`-deleteUserGroup <name>`

Removes a user group by name.

`-displayConfigReport [<report>[, <report>]]`

Displays security configuration reports. Available reports are: `unusualRoleCombinations, emptySecurityGroups, emptyUserGroups, securityGroupsWithSameName, usersWithoutGroups, tenantsWithSameName, usersWithoutRoles`

If no reports are specified, all available reports are run.

`-listNode <node name>`

Displays the UUIDs of the security group and tenant associated with the specified node. The node can be specified as name, hostname, or UUID. The output lists node UUID and name; security group UUID and name; and tenant UUID and name on separate lines.

`-listNodesInSecurityGroup <groupName or uuid>`

Lists nodes in a security group by security group name or UUID.

`-listSecurityGroupForTenant <uuid>`

Displays the configured default security group for the specified tenant.

`-listSecurityGroups`

Lists the names of all configured security groups.

`-listTenants`

Lists the names of all configured tenants.

`-listUserGroupMembers <groupName>`

Lists users in the specified user group.

`-listUserGroups <user>`

Lists user group membership for the specified user.

`-listUserGroupsForSecurityGroup <groupName or uuid>`

Lists user groups associated with the specified security group.

`-removeUserFromGroup (-user <name> -userGroup <name>) | -file <filename>`

Removes mappings between user accounts and user groups.

`-user`

The username of the user account to modify.

`-userGroup`

The name of the user group to unmap from the specified user account.

`-file`

Path to a CSV-formatted file containing lists of user to user group mappings with the format: `user, userGroup`

`-deleteUserGroup <name> | -file <name>`

Removes user groups by name. Mappings between the user group and user accounts and security groups are also removed.

`-file`

Path to a CSV-formatted file containing lists of user to user group mappings with the format: `usergroup, description`. This format is the same as createUserGroup; however, only the name is used to match the groups to be removed.

`-removeUserGroupFromSecurityGroup (-userGroup <groupName> -securityGroup <groupName or uuid> [-role <role>]) | -file <name>`

Removes mappings between user groups and security groups.

`-userGroup`

The name of the user group.

`-securityGroup`

The name or UUID of the security group.

`-role`

An optional role. If no role is specified, mappings for all roles are removed.

`-file`

Path to a CSV-formatted file containing lists of user to user group mappings with the format: `userGroup, securityGroup, role`

`-updateUserGroup <name> ([-displayName <user friendly group name>] [-description <description>] [-directoryServiceName <dn>]) | -file <name>`

Updates a user group. All user group attributes except name can be updated.

`-displayName`

Optional friendly name for the user group.

`-description`

Optional description of the group.

`-directoryServiceName`

Optional for directory service users. Use this option to pair a directory service distinguished name with this user group.

`-file`

Path to a CSV-formatted file containing lists of user groups with the format: `name, displayName, description, directoryServiceName`

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost <hostname>`

The server jndi host; default is localhost.

`-jndiPort <port>`

The server jndi port; default is 1099.

EXAMPLES

```
nnmsecurity.ovpl -createTenant myTenant
```

Creates a tenant named myTenant.

```
nnmsecurity.ovpl -listTenants
```

Lists all configured tenants.

```
nnmsecurity.ovpl -createTenant "Tenant with a space",nnmsecurity.ovpl -createTenant
\!Tenant
```

Depending on the shell you use, you can use quotation marks around the tenant name to create a tenant with spaces in the name, or you can use the escape character to create a tenant with a special character in its name.

```
nnmsecurity.ovpl -createSecurityGroup mySecurityGroup
```

Creates the security group `mySecurityGroup`.

```
nnmsecurity.ovpl -createSecurityGroup "Group with a space",nnmsecurity.ovpl -
createSecurityGroup \!MyGroup
```

Depending on the shell you use, you can use quotation marks around the security group name to create a security group with spaces in the name, or you can use the escape character to create a security group with a special character in its name.

```
nnmsecurity.ovpl -listSecurityGroups
```

Lists all configured security groups.

```
nnmsecurity.ovpl -listNode myNode
```

Lists the associated security group and tenant for the supplied node.

DIAGNOSTICS

`nnmsecurity.ovpl` returns the following exit codes:

`0`

Operation was successful.

`1`

An error occurred; see error message for details.

AUTHOR

`nnmsecurity.ovpl` was developed by Hewlett-Packard Company.

FILES

The environment variable below represents a universal path that is established according to your shell and platform requirements:

Windows: `%NNM_BIN%\nnmsecurity.ovpl`

UNIX: `$NNM_BIN/nnmsecurity.ovpl`

SEE ALSO

nnm.properties(4).

Return to Reference Pages Index

Name

nnmseeddelete.ovpl — Remove seed from the NNM topology database

SYNOPSIS

`nnmseeddelete.ovpl` -help | -seed <*`seed`*> | [-u <*`username`*> -p <*`password`*>] [-jndiHost <*`hostName`*> Default: localhost] [-jndiPort <*`port`*> Default: 1099]

DESCRIPTION

`nnmseeddelete.ovpl` removes a seed from the system.

Parameters

`nnmseeddelete.ovpl` supports the following options:

`-seed <`*`seed`*`>`

The seed to delete. A seed can be a host name or an IP address and must match exactly as listed in the seed list.

`-u <`*`username`*`>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <`*`password`*`>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost <`*`serverName`*`>`

The server JNDI host; default is `localhost`.

`-jndiPort <`*`port`*`>`

The server JNDI port; default is `1099`.

EXAMPLES

`nnmseeddelete.ovpl -u username -p password -seed 10.1.2.3`

Removes the seed `10.1.2.3`. (You must provide an NNMi administrator username and password.)

Diagnostics

`nnmseeddelete.ovpl` returns the following exit codes:

`0`

Operation was successful.

`1`

An error occured; see error message for details.

`-help`

Prints the usage statement.

AUTHOR

`nnmseeddelete.ovpl` was developed by Hewlett-Packard Company.

FILES

`$NNM_BIN``/nnmseeddelete.ovpl`

SEE ALSO

nnmloadseeds.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmsetdampenedinterval.ovpl — sets the dampened interval for all incident configurations

SYNOPSIS

`nnmsetdampenedinterval.ovpl` [[-hours *`hours`*] [-minutes *`minutes`*] [-seconds *`seconds`*] [-u *`username`*] [-p *`password`*]]

DESCRIPTION

`nnmsetdampenedinterval.ovpl` Sets the dampened interval for all incident configurations. The maximum dampened interval that can be set is 60 minutes. A dampened interval of at least 6 minutes is recommended. At least one of hours, minutes, or seconds must be specified. To disable dampening, set hours, minutes, and seconds to 0.

Parameters

`-hours` *`hours`*

Specify the number of hours for the dampened interval. If specified, the value for *`hours`* must be greater than or equal to `0`.

`-minutes` *`minutes`*

Specify the number of minutes for the dampened interval. If specified, the value for *`minutes`* must be greater than or equal to `0`.

`-seconds` *`seconds`*

Specify the number of seconds for the dampened interval. If specified, the value for *`seconds`* must be greater than or equal to `0`.

`-jndiHost` *`hostname`*

Server `jndi` host. Default is `localhost`.

`-jndiPort` *`port`*

Server `jndi` port. Default is `1099`.

`-u <`*`username`*`>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <`*`password`*`>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

EXAMPLES

Set the dampened interval for all incident configurations to 10 minutes 10 seconds:

```
nnmsetdampenedinterval.ovpl -hours 0 -minutes 10 -seconds 10
```

Set the dampened interval for all incident configurations to 1 hour:

```
nnmsetdampenedinterval.ovpl -hours 1
```

Set the dampened interval for all incident configurations to 6 minutes:

```
nnmsetdampenedinterval.ovpl -minutes 6
```

Set the dampened interval for all incident configurations to 30 seconds:

```
nnmsetdampenedinterval.ovpl -seconds 30
```

Set the dampened interval for all incident configurations to 10 minutes 10 seconds:

```
nnmsetdampenedinterval.ovpl -minutes 10 -seconds 10
```

Disable dampening for all incident configurations:

```
nnmsetdampenedinterval.ovpl -hours 0 -minutes 0 -seconds 0
```

AUTHOR

`nnmsetdampenedinterval.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnm.properties(4)

Return to Reference Pages Index

Name

nnmsetiospeed.ovpl — Set the input or output speed on interfaces

SYNOPSIS

`nnmsetiospeed.ovpl`
{ -?
[-node
| -interface <*`interface ID list`*> | -input <*`The input speed to set`*> | -output <*`The output speed to set`*>] [-file <*`The CSV file to process`*>]} [-verbosity *`the verbosity of the command`*] [-failfast *`The fail-fast behavior of the command`*]
[-jdniHost *`hostname`*]
[-jdniPort *`port`*]

[-u *`username`* -p *`password`*]

DESCRIPTION

`nnmsetiospeed.ovpl` will set the input or output speed of the specified interfaces on the specified nodes. Speeds are suffixed by the units desired, so 10mb is equivalent to 10000kb. If fail-fast behavior is desired, the command will not make any modifications if any errors are encountered during execution.

Parameters

`nnmsetiospeed.ovpl` supports the following options:

`-file <file path>`

A file containing a comma-delineated list of commands to process

`-node <node name, long name, UUID, or IP address (v4 or v6)>`

Change the input or output speed of the specified interfaces on the indicated node. The `$ALL` option will process all the interfaces on the indicated node. Nodes may be identified by short name, long name, UUID, or IP address. The $ALL macro may be used in batch mode as well.

Optionally one of the following parameters can be specified, only the specified attribute will be printed:

`-failfast`

Ensures that no changes will be made if an error is encountered

`-verbosity`

Sets the verbosity of the command. The value may be one of "silent", "verbose", or "normal". When verbosity is set to "silent," no output is provided. When verbosity is set to "verbose,"

every error and status message is displayed. Normal (the default value), provides output somewhere between.

`-uuid`

Prints the UUID of each node belonging to the node group.

`-?`

Prints the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost`

The server JNDI host; default is`localhost`.

`-jndiPort`

The server JNDI port; default is`1099`.

Syntax of Comma-Separated File

The CSV file you supply must have the following syntax:

Required. Column 1. You must provide a value for the node identifier

Required. Column 2. You must provide a value for interface identifiers

Requred. One of Column 2 or Column 3 (input/output speed)

An input or output speed can be omitted by inserting nothing at the position in question

- Column 1(A) : Node Identifier

  Required. The short name, long name, UUID, or IP address of the node to set the speed of interfaces on

- Column 2(B) : Interface Identifier List

  Required. A semicolon-delineated list of interface identifiers or $ALL macro. Interface IDs may be the interface index, alias, name, or description of the interface. The list of interface identifiers should be enclosed in quotation marks to prevent the shell or console from prematurely terminating the command.

- Column 3(C) : Input speed

Optional (if output speed is specified). The input speed of the interfaces to set

- Column 3(C) : Output speed

  Optional (if output speed is specified). The input speed of the interfaces to set

EXAMPLES

To set the speed of all the interfaces on a node with the name `example-node` to 10 MB/s

```
nnmsetiospeed.ovpl -node example-node -interface $ALL -input 10mb -output 10mb
```

To set the speed of all the interface with name, description, alias, and index (respectively) `ifname,ifdescr,ifalias,10` a node with the name `example-node` to 1 GB/s

```
nnmsetiospeed.ovpl -node example-node -interface "ifname;ifdescr;ifalias;10" -input 10mb
-output 1gb
```

RETURN VALUE

`nnmsetiospeed.ovpl` exits with the status `0` (zero) if no errors were encountered, 1 otherwise.

AUTHOR

`nnmsetiospeed.ovpl` was developed by Hewlett-Packard Company.

FILES

```
$NNM_BIN/nnmsetiospeed.ovpl
```

Return to Reference Pages Index

Name

nnmsetofficialfqdn.ovpl — Set the official Fully Qualified Domain Name (FQDN) of the NNMi management server.

SYNOPSIS

`nnmsetofficialfqdn.ovpl` [-f | -force] <*`fqdn`*>

DESCRIPTION

Use the `nnmsetofficialfqdn.ovpl` script to change the official FQDN for the NNMi management server after installation. If you run this script without any arguments, it sets the official FQDN to the default value. The default value is obtained by performing a hostname lookup. See the reference page for the `nnmofficialfqdn.ovpl` script using the -m option for more information about the default hostname lookup.

If the FQDN is changed, the user is prompted to auto generate a new SSL certificate. All certificates that NNMi auto generates have the following alias syntax: `fqdn.selfsigned`. Because the SSL certificate is tied to the FQDN, HTTPS breaks if a new certificate is not generated and added to both the `nnm.keystore` and `nnm.truststore` files. If you provide the `-force` option, the `nnmsetofficialfqdn.ovpl` script generates and inserts the certificate into both the `nnm.keystore` and `nnm.truststore` files. If the new certificate's alias already exists in either the `nnm.keystore` or `nnm.truststore` files, the certificate is not inserted.

If a new certificate is generated, the system property `com.hp.ov.nms.ssl.KEY_ALIAS` in the `nms-local.properties` file is updated to reference the new certificate alias. This is a necessary step for HTTPS to work correctly when NNMi is started again.

Options

There are two optional arguments to `nnmsetofficialfqdn.ovpl`

`<fqdn>`

Resets the official FQDN to <fqdn> on confirmation from the user.

If the value of <fqdn> does not have a domain (for example, it is a short hostname, IP address or localhost), SSO is disabled.

`-force|-f`

This is a *force flag* that does not prompt the user for confirmation and does not print verbose information. When used alone, the force flag sets the official FQDN to the default. When *`fqdn`* is used after this flag, it changes the official FQDN to *`fqdn`*.

EXAMPLES

Set the FQDN to the default value:

```
nnmsetofficialfqdn.ovpl
```

Set the FQDN to the value somehost.somedomain:

```
nnmsetofficialfqdn.ovpl somehost.somedomain
```

Force the official FQDN to the default value:

```
nnmsetofficialfqdn.ovpl -f
```

AUTHOR

`nnmsetofficialfqdn.ovpl` was developed by Hewlett-Packard Company.

FILES

`nnmsetofficialfqdn.ovpl` resides in the `%NNM_BIN%` (Windows) or `$NNM_BIN` (UNIX) directory.

SEE ALSO

nnmofficialfqdn.ovpl(1M), nnmsso.ovpl(1M).

Return to Reference Pages Index

Name

disco.NoVLANIndexing — Specifies certain nodes where VLAN Indexing should be skipped during discovery polling.

SYNOPSIS

`disco.NoVLANIndexing`

DESCRIPTION

One of the methods NNMi uses to learn layer 2 connectivity between and among switch devices in a managed network is to retrieve the `dot1dTpFdbTable` (FDB) from the switches. However, for Cisco switches, NNMi must use a `VLAN-indexing` method to retrieve the entire FDB. Using this method, NNMi retrieves the FDB once for each configured VLAN on the Cisco device. If there is a large number of VLANs configured on each device, retrieving the FDB with `VLAN-indexing` might take a very long time, sometimes even hours, to complete.

Cisco switches are often configured to use the Cisco Discovery Protocol (CDP). CDP is considered to be a superior method for learning layer 2 connectivity. Large switches located in the in the core of the network might contain many VLANs. These switches typically do not have end nodes connected directly to them. If the switches you want to manage do not have end nodes connected directly to them, you might want to suppress the collection of the FDB on these large switches. NNMi still completes the layer 2 discovery using data collected from CDP. These large switches are prime candidates for suppression of `VLAN-indexing`. Do not suppress `VLAN-indexing` on smaller switches located at the network's edge (often known as access switches) that have many end nodes attached to them.

You can configure NNMi to suppress VLAN indexing. To do this, the NNMi administrator needs to create the `disco.NoVLANIndexing` file, where the name of the file is case-sensitive. The `ovjboss` service reads the `disco.NoVLANIndexing` file when it starts. If the NNMi administrator makes changes to the `disco.NoVLANIndexing` file after the `ovjboss` service starts, those changes will not take effect until the next time the `ovjboss` service starts. By default, the `disco.NoVLANIndexing` file does not exist. If the `disco.NoVLANIndexing` does not exist, this feature is disabled and NNMi attempts to use `VLAN-indexing` to collect the entire FDB table on all devices.

The `disco.NoVLANIndexing` file can contain IP addresses, IP address ranges, and comments. A comment consists of the pound (or hash) sign (`#`) and all characters between `#` and the end of the line. NNMi treats an empty line as a comment. IP addresses are specified in the standard IP version 4 dotted-decimal notation or standard IP version 6 format (RFC 2373).

For details on the format of IP address ranges, see the *Configure Address Ranges for Regions* section of the NNMi help.

NNMi considers a node to match if one of the listed IP addresses matches a node's management address. Other IP addresses hosted by the node are not considered. If a node matches one of the addresses in the `disco.NoVLANIndexing` file, NNMi collects only the default FDB (the FDB which is accessible by using the community string with no @vlan-id suffix appended).

Disabling the collecting of the entire FDB might cause some inaccuracies in the layer 2 layout of the

managed network. HP is not responsible for these inaccuracies. Carefully consider which switches you include in the `disco.NoVLANIndexing` file.

EXAMPLES

The following is an example of a `disco.NoVLANIndexing` file:

```
#This entry suppresses VLAN-indexing for the node whose management address is
10.2.37.149
10.2.37.149

192.168.100-101.1 #This entry causes the nodes 192.168.100.1 and 192.168.101.1 to be
skipped, too

# Here are some examples of IPv6 addresses and ranges:
   2136::8:800:200C:417a
   fd01::a352:1245:fc4B
   2001:D88:2:0:a07:ffff:0a01:3200-37ff
```

AUTHOR

`disco.NoVLANIndexing` was developed by Hewlett-Packard Company.

FILES

```
$NnmDataDir/shared/nnm/conf/disco/disco.NoVLANIndexing

%NnmDataDir%\shared\nnm\conf\disco\disco.NoVLANIndexing
```

SEE ALSO

See the *Maintaining NNMi* chapter in the newest version of the *NNMi Deployment Reference* for more information.

See the *Configure Address Ranges for Regions* section of the NNMi help.

Return to Reference Pages Index

Name

disco.SkipXdpProcessing — Contains a list of management IP addresses for nodes NNMi should not query for discovery protocol information.

SYNOPSIS

`disco.SkipXdpProcessing`

DESCRIPTION

One method NNMi uses to discover layer 2 connectivity between and among network devices in a managed network is to collect information from the devices related to their discovery protocols. There are many defined discovery protocols. For example, Link Layer Discovery Protocol (LLDP) is an industry standard protocol, while there are many vendor-specific protocols like Cisco Discovery Protocol (CDP) for Cisco devices. These are all handled by NNMi discovery in the `XdpAnalyzer`.

You can configure NNMi to suppress discovery protocol collections for devices you specify. This feature makes use of a configuration file, `disco.SkipXdpProcessing`, that the NNMi administrator creates. The name of the file is case-sensitive. The `ovjboss` service reads the `disco.SkipXdpProcessing` when it starts up. If the NNMi administrator makes changes to this file after the `ovjboss` service starts up, those changes will not take effect until the next time the `ovjboss` service starts. By default, the `disco.SkipXdpProcessing` file does not exist. If the `disco.SkipXdpProcessing` does not exist, this feature is disabled and NNMi attempts to collect discovery protocol information from all managed nodes.

For more information about the known problems fixed by this feature, refer to the SEE ALSO section below.

The `disco.SkipXdpProcessing` file can contain IP addresses and comments. A comment consists of the pound (or hash) sign (`#`) and all characters between `#` and the end of the line. NNMi treats an empty line as a comment. Specify IP addresses in the standard IP version 4 dotted-decimal notation or standard IP version 6 format (RFC 2373).

NNMi considers a node to match if one of the listed IP addresses matches a node's management address. Other IP addresses hosted by the node are not considered. If a node matches one of the addresses in the `disco.SkipXdpProcessing` file, NNMi skips the `XdpAnalyzer` service for that node and does not collect discovery protocol information.

Disabling the discovery protocol processing of a node or nodes might cause some inaccuracies in the layer 2 layout of the managed network. HP is not responsible for these inaccuracies.

EXAMPLES

The following is an example of a `disco.SkipXdpProcessing` file:

```
#This entry supresses the XdpAnalyzer processing for the node whose management address
is 10.2.37.149
10.2.37.149
```

```
192.168.100.1 #This entry causes the node 192.168.100.1 to be skipped, too

# Here are some examples of IPv6 addresses:
   2136::8:800:200C:417a
   fd01::a352:1245:fc4B
```

AUTHOR

`disco.SkipXdpProcessing` was developed by Hewlett-Packard Company.

FILES

`$NnmDataDir/shared/nnm/conf/disco/disco.SkipXdpProcessing`

`%NnmDataDir%\shared\nnm\conf\disco\disco.SkipXdpProcessing`

SEE ALSO

See the *Maintaining NNMi* chapter in the newest version of the *NNMi Deployment Reference* for more information.

Return to Reference Pages Index

Name

hostnolookup.conf — file containing hostnames or hostname wildcards that should not be resolved to IP addresses using the system IP name server

SYNOPSIS

`hostnolookup.conf`

DESCRIPTION

`hostnolookup.conf` is a file used by the `ovjboss` process to determine whether a hostname should be resolved to an IP address using the system IP name server. The `ovjboss` process attempts to match a hostname against each entry in the `hostnolookup.conf` file before attempting to resolve the hostname to an IP address. If a match is found, the `ovjboss` process does not attempt to resolve the hostname to an IP address using the system IP name server.

Add entries to the file containing one hostname or hostname wildcard. Each entry must be on a single line. To add comments, place a number sign (`#`) in front of the comment. That causes the remainder of the line to be ignored. You can add blank lines to the `hostnolookup.conf` file.

Use the `hostnolookup.conf` file if you determine that a specific hostname (or set of hostnames) cannot be resolved to an IP address using the systems IP name server.

The administrator must create the `hostnolookup.conf` file. It does not exist by default.

If you modify the `hostnolookup.conf` file while the `ovjboss` process is running, use the `$NnmInstallDir/support/nmsdnssync.ovpl` script to load the updated file. The `nmsdnssync.ovpl` script also reloads the `ipnolookup.conf` file.

EXAMPLES

The following is an example of a `hostnolookup.conf` file:

```
# A single hostname
badsys.mydomain.mycorp.com
# An IP wildcard
*.baddomain.mycorp.com
```

In the first example, the system name is bad in some way, causing some DNS servers to respond with unexpected results. In the second example, there is a domain that cannot be resolved. Adding these entries to the `hostnolookup.conf` file stops NNMi from attempting to resolve the hostnames.

AUTHOR

`hostnolookup.conf` was developed by Hewlett-Packard Company.

FILES

`%NNM_DATA%\shared\nnm\conf\hostnolookup.conf`

`$NNM_DATA/shared/nnm/conf/hostnolookup.conf`

SEE ALSO

ipnolookup.conf(4).

Return to Reference Pages Index

Name

ipnolookup.conf — file containing IP addresses or IP wildcards that should not be resolved to hostnames using the system IP name server

SYNOPSIS

`ipnolookup.conf`

DESCRIPTION

`ipnolookup.conf` is a file used by all NNMi processes to determine whether an IP address should be resolved to a hostname using the system IP name server. NNMi processes attempt to match an IP address against each entry in the `ipnolookup.conf` file before attempting to resolve the IP address to a hostname. If a match is found, the NNMi process does not attempt to resolve the IP address to a hostname using the system IP name server.

Add entries to the file containing one IP address or IP wildcard per line. Each entry must be on a single line. To add comments, place a number sign (`#`) in front of the comment. This causes the remainder of the line to be ignored. You can add blank lines to the `ipnolookup.conf` file.

Use the `ipnolookup.conf` file when you determine that a specific IP address (or range of IP addresses) cannot be resolved to a hostname using the system IP name server.

The administrator must create the `ipnolookup.conf` file. It does not exist by default.

If you modify the `ipnolookup.conf` file while NNMi processes are running, run the `$NnmInstallDir/support/nmsdnssync.ovpl` script with no arguments to load the modifications you made to the `ipnolookup.conf` file.

EXAMPLES

The following is an example of a `ipnolookup.conf` file:

```
# A single IP address
192.168.1.100
# An IP wildcard
10.*.*.*
# An IP wildcard range
192.168.1.101-255
```

In the first example, the single IP address could be routed to the Internet because many web sites use a `192.168.*.*` IP address. In the second example, the IP wildcard range could be NAT addresses. As such, they are not suitable for communications. In the third example, the IP wildcard range could be a set of addresses used for some purpose other than the primary IP address.

AUTHOR

`ipnolookup.conf` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_DATA%\shared\nnm\conf\ipnolookup.conf`

UNIX: `$NNM_DATA/shared/nnm/conf/ipnolookup.conf`

SEE ALSO

hostnolookup.conf(4).

Return to Reference Pages Index

Name

macdedupexceptions.txt — file containing sysObjectId values for types of nodes that are to be considered exceptions to the mac-address-based node deduplication logic

SYNOPSIS

`macdedupexceptions.txt`

DESCRIPTION

NNMi uses a number of complex algorithms to detect that a node being processed is really a duplicate of another node in the database. In some cases, NNMi compares MAC addresses to try to determine if a node has received a new IP address due to a DHCP lease expiring. This might cause issues with some network devices such as firewalls and loadbalancers. There are some cases where these devices might use common IP addresses and MAC addresses across multiple distinct devices. Normally, NNMi distinguishes between these devices by seeing that the SNMP sysName is different. However, there are also cases where the SNMP sysName can not be made different. In these cases, NNMi might delete one of the devices from the database claiming that it is a duplicate.

For devices such as these loadbalancers and firewalls, NNMi can be told to modify its de-duplication algorithm by listing these devices' SNMP sysObjectId values in the `macdedupexceptions.txt` file. Devices that are good candidates for inclusion in this file have the following characteristics:

- The device must not obtain its IP address from a DHCP server. Its IP address should be statically assigned.
- The device must use a unique management IP address.

The following are examples of devices where this configuration file can prove useful:

- The device is configured in a redundant configuration with another device that uses some common IP and MAC addresses, and shares a common SNMP sysName.
- The device is a physical device that supports several virtual instances, where each instance might be using similar IP and MAC addresses, and sharing a common SNMP sysName.

HP recommends that the NNMi administrator only add entries to this file if they are needed to have devices properly discovered. Adding entries which are not needed may cause unexpected results.

The file can contain one or more SNMP sysObjectId values, one value per line. Lines starting with a `#` are treated as comment lines, as are blank lines. Also, a comment can follow a sysObjectId, starting with a `#` to the end of the line. White-space in front of or following a sysObjectId is ignored. And a leading dot (`.`) on the sysObjectId value is optional.

This file does not exist by default. If it is needed, the NNMi administrator must create it. The file is read by NNMi at startup time. Any changes made after NNMi starts will not be active until NNMi is re-started.

EXAMPLES

The following is an example of a `macdedupexceptions.txt` file:

```
# F5 BIG-IP Pb200 loadbalancer device
.1.3.6.1.4.1.3375.2.1.3.4.19

1.3.6.1.4.1.9.1.1291  #Cisco ACE Service Module
```

AUTHOR

`macdedupexceptions.txt` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_DATA%\shared\nnm\conf\disco\macdedupexceptions.txt`

UNIX: `$NNM_DATA/shared/nnm/conf/disco/macdedupexceptions.txt`

Return to Reference Pages Index

Name

nnm.ports — The following information shows the ports the NNMi management server listens on. In the case of port conflicts, you can change many of these port numbers.

SYNOPSIS

`nnm.ports`

There is no command synopsis. The *nnm.ports* reference page documents the well-known ports you can change by modifying the `nms-local.properties` file.

DESCRIPTION

To change these port numbers, follow these steps:

1. Edit the `%NnmDataDir%\conf\nnm\props\nms-local.properties` (Windows) or `/var/opt/OV/conf/nnm/props/nms-local.properties` (Unix) file to modify the ports used by NNMi.

2. Identify the line containing the port number you need to change.

3. If necessary, remove the `#!` characters at the beginning of the line.

4. Modify the port number; then save your changes.

5. From a command prompt, run `ovstop`, then `ovstart` to restart NNMi.

The following ports are currently defined:

nmsas.server.port.web.http=80

> This TCP port is used as the default HTTP port for Web UI and Web Services. The installation script prompts you to set this value during the NNMi installation. You can change this port value by modifying the `nms-local.properties` file or during NNMi installation.

nmsas.server.port.web.https=443

> This TCP port is used as the default secure HTTPS port (SSL) for Web UI and Web Services. The installation script prompts you to set this value during the NNMi installation.

nmsas.server.port.naming.rmi=1098

> This is the default TCP port of the RMI naming service.

nmsas.server.port.remoting.ejb3 =1099

> This TCP port is the default listening port for the bootstrap JNP service (JNDI provider).

nmsas.server.port.remoting.ejb3=3873

This TCP port is used for remote access by command line tools running on the global NNMi management server.

nmsas.server.port.jmx.jrmp=4444

This TCP port is the RMI Object port (Java Remote Method Protocol) used by RMI to transfer data (JRMP invoker).

nmsas.server.port.jmx.rmi=4445

This TCP port is the default port (invoker port) used when pooling RMI requests.

nmsas.server.port.invoker.unified=4446

This TCP port is the default RMI remoting server connector port. The jboss `Remoting` service uses this port.

nmsas.server.port.hq=4457

This TCP port is used for un-encrypted Global Network Management traffic.

nmsas.server.port.hq.ssl=4459

This TCP port is used for encrypted Global Network Management traffic.

nmsas.server.port.ts.recovery=4712

This TCP port is an internal transaction service port used by NNMi.

nmsas.server.port.ts.status=4713

This TCP port is an internal transaction service port used by NNMi.

nmsas.server.port.ts.id=4714

This TCP port is an internal transaction service port used by NNMi.

com.hp.ov.nms.postgres.port=5432

This TCP port is the port the embedded database listens on for this NNMi management server (PostgreSQL port).

AUTHOR

`nnm.ports` was developed by Hewlett-Packard Company.

FILES

Windows: `%NnmDataDir%\conf\nnm\props\nms-local.properties`

UNIX: `/var/opt/OV/conf/nnm/props/nms-local.properties`

SEE ALSO

ovstatus(1), ovstart(1M), ovstop(1M).

Return to Reference Pages Index

Name

nnm.properties — file containing the username and password for command line tools

SYNOPSIS

`nnm.properties`

DESCRIPTION

If you frequently run NNMi command line tools, HP recommends you create an `nnm.properties` file. This file contains a username and password that you can use in place of the `-u` and `-p` command line options. Using the `nnm.properties` file, you can run many commands without entering your password. The `nnm.properties` file contains an encrypted version of the password and should be created by running the `nnmsetcmduserpw.ovpl` script. Never manually edit the `nnm.properties` file. The `nnmsetcmduserpw.ovpl` script creates or edits a file placed in an `.nnm` subdirectory under your home directory at the following locations:

- *Windows*: *`drive`*`:\Documents and Settings\`*`username`*`\.nnm\`
- *UNIX*: `~/.nnm/`

The contents of this file must be defined as follows:

`nnm.username`

The account user name.

`nnm.password`

The account encrypted password.

Most command line tools require an administrative user with an *`admin`* role assigned to the tool.

AUTHOR

`nnm.properties` was developed by Hewlett-Packard Company.

FILES

The environment variable below represents a universal path that is established according to your shell and platform requirements:

Windows: *`drive`*`:\Documents and Settings\`*`username`*`\.nnm\nnm.properties`

UNIX: *`$HOME`*`/.nnm/nnm.properties`

SEE ALSO

nnmsetcmduserpw.ovpl(1).

Return to Reference Pages Index

Name

ldap.properties — file containing the settings for communicating with and building LDAP queries to the directory service

SYNOPSIS

`ldap.properties`

DESCRIPTION

For detailed information about the contents of the `ldap.properties` file, see the *Integrating NNMi with a Directory Service through LDAP* chapter of the *NNMi Deployment Reference*.

After modifying this file, run the `nnmldap.ovpl -reload` command to reload the configuration.

AUTHOR

`ldap.properties` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_SHARED_CONF%\ldap.properties`

UNIX: `$NNM_SHARED_CONF/ldap.properties`

SEE ALSO

nnmldap.ovpl(1M).

Return to Reference Pages Index

Name

nnmtrapd.conf — Filter file to block traps based on IP address or OID

SYNOPSIS

`nnmtrapd.conf`

DESCRIPTION

`nnmtrapd.conf` file can be used to configure filters to block traps based on both IP address and trap OID.

The filters are entered one per line. Each filter consists of an IP address, range or wildcard folowed by one or more comma separated list of trap OIDs or range of trap OIDs or wildcards.

The format of the filter is:

```
<IP Address, OID[,OID]*>
```

IP Address could be a single IP Address or a pattern in prefix/prefix-length notation or range-wildcard notation. The special notation of `"*"` indicates all addresses. You may not combine the prefix/prefix-length notation with the range-wildcard notation in the same address. Host name cannot be specified instead of an address. Every filter entry should have a unique address (single, wild card or range). Examples of addresses in prefix/prefix-length notation are:

```
10.2.112.0/20
1080:0:a00::/44
```

The same addresses in range-wildcard notations are:

```
10.2.112-127.*
1080:0:a00-a0f:*:*:*:*:*
```

The trap OID can also be specified as a range or as wildcard. However, you may only use either a range or a wildcard in one OID. Only the last sub OID of an OID can be specified as a wildcard or range. The special notation `".*"` indicates all OIDs. Some examples are:

```
.1.3.6.1.4.1.11.2.17.1.0.58915834-58915868
.1.3.6.1.4.1.11.*
```

For generic traps like link up, you can append the enterprise OID of the vendor to the trap OID for blocking a specific vendor. Conversely, to block a generic trap from all vendors you need to append a wildcard to the trap OID.

Blocking all traps from all address is not allowed. So the following entry is ignored:

```
<*, .*>
```

EXAMPLES

The following example blocks all generic traps from all subnets in the range 10.2.120 to 10.2.127:

```
<10.2.120.0/21, .1.3.6.1.6.3.1.1.5.*>
```

The following example blocks link up traps from all devices in the 10.6.112/21 subnet whose enterprise OID is .1.3.6.1.4.1.11.2.3.7.11.17:

```
<10.6.112.0/21, .1.3.6.1.6.3.1.1.5.4.1.3.6.1.4.1.11.2.3.7.11.17>
```

The following example blocks link up traps from all devices in the 10.6.112/21 subnet.

```
<10.6.112.0/21, .1.3.6.1.6.3.1.1.5.4.*>
```

The following example blocks all traps under the OID .1.3.6.1.4.1.11.2.17 and the authentication failure trap from a single IPv6 address.

```
<1080::8:800:200c:417a, .1.3.6.1.4.1.11.2.17.*, .1.3.6.1.6.3.1.1.5.5.*>
```

AUTHOR

`nnmtrapd.conf` was developed by Hewlett-Packard Company.

FILES

```
$NnmDataDir/shared/nnm/conf/nnmtrapd.conf
```

SEE ALSO

nnmtrapconfig.ovpl(1M).

Return to Reference Pages Index

Name

trapFilter.conf — Filter file to block traps based on IP address or OID

SYNOPSIS

`trapFilter.conf`

DESCRIPTION

Use the `trapFilter.conf` file to configure filters to block traps based on both IP address and trap OID. This is similar to using the `nnmtrapd.conf` file, except that traps blocked by the `trapFilter.conf` file do not get stored in the trap binary store, nor are they used to analyze trap rates. Trap rates are not affected by incoming traps blocked by the `trapFilter.conf`. file. NNMi does not store incoming traps blocked by the `trapFilter.conf` file.

Enter the filters one filter per line. Each filter consists of an IP address, range, or wildcard followed by one or more comma separated trap OIDs, range of trap OIDs, or wildcards.

The format of the filter is as follows:

```
<IP Address, OID[,OID]*>
```

`IP Address` could be a single IP Address or a pattern in prefix/prefix-length notation or range-wildcard notation. The special notation of `"*"` indicates all addresses. Do not combine the prefix/prefix-length notation with the range-wildcard notation in the same address. Do not specify the hostname instead of an address. Every filter entry should have a unique address (single, wildcard, or range). Examples of addresses in prefix/prefix-length notation are as follows:

```
10.2.112.0/20
1080:0:a00::/44
```

The same addresses in range-wildcard notations are:

```
10.2.112-127.*
1080:0:a00-a0f:*:*:*:*:*
```

Specify the trap OID as a range or as wildcard. Only use a range or a wildcard in one OID. Only specify the last sub OID of an OID as a wildcard or range. The special notation `".*"` indicates all OIDs. Some examples are as follows:

```
.1.3.6.1.4.1.11.2.17.1.0.58915834-58915868
.1.3.6.1.4.1.11.*
```

For generic traps like `linkUp`, you can append the enterprise OID of the vendor to the trap OID for blocking a specific vendor. Conversely, to block a generic trap from all vendors you need to append a wildcard to the trap OID.

Blocking all traps from all addresses is not allowed. So the following entry is ignored:

```
<*, .*>
```

EXAMPLES

The following example blocks all generic traps from all subnets in the range 10.2.120 to 10.2.127:

```
<10.2.120.0/21, .1.3.6.1.6.3.1.1.5.*>
```

The following example blocks link up traps from all devices in the 10.6.112/21 subnet whose enterprise OID is .1.3.6.1.4.1.11.2.3.7.11.17:

```
<10.6.112.0/21, .1.3.6.1.6.3.1.1.5.4.1.3.6.1.4.1.11.2.3.7.11.17>
```

The following example blocks link up traps from all devices in the 10.6.112/21 subnet.

```
<10.6.112.0/21, .1.3.6.1.6.3.1.1.5.4.*>
```

The following example blocks all traps under the OID .1.3.6.1.4.1.11.2.17 and the authentication failure trap from a single IPv6 address.

```
<1080::8:800:200c:417a, .1.3.6.1.4.1.11.2.17.*, .1.3.6.1.6.3.1.1.5.5.*>
```

AUTHOR

`trapFilter.conf` was developed by Hewlett-Packard Company.

FILES

```
$NnmDataDir/shared/nnm/conf/trapFilter.conf
```

SEE ALSO

nnmtrapconfig.ovpl(1M).

Return to Reference Pages Index

Name

hosted-object-trapstorm.conf — Configuration file for hosted object trap storm detection and suppression. Block sets of traps by a hosted object's overall trap rate.

SYNOPSIS

`hosted-object-trapstorm.conf`

DESCRIPTION

Use the `hosted-object-trapstorm.conf` file to configure filters to block traps from hosted objects. Configure groups of trapOID to varbindOID pairs for blocking. When a trap is processed by this filter, the trap's source device is resolved by the configured varbindOID. The statistical tracker for the resolved hosted object is checked to ensure that its trap rate is below a configurable threshold. If the trap rate is above the threshold, the trap is dropped.

A message is logged to `nnm-trace.#.#.log` when a trap is dropped. Traps blocked by the `hosted-object-trapstorm.conf` file do not get stored in the trap binary store. Traps blocked by the `hosted-object-trapstorm.conf` file do not affect trap rates. NNMi does not store traps blocked by the `hosted-object-trapstorm.conf` file.

Configuration blocks are used to accomplish configuration, and can span multiple lines. Configuration blocks consist of the following comma separated values: GroupID, TrapOID, VarbindOID.

The format of a configuration block is as follows:

```
<GroupID, TrapOID, VarbindOID [,TrapOID, VarbindOID]*>
```

The following rules govern `configuration blocks`:

```
1. All entries are separated by ","
2. Never start a configuration line with a ","
3. If a configuration line does not end with a "," then a comma is implied.
4. Any text following "#" is considered a comment and will not be parsed.
5. Comments are allowed within the configuration block.
6. Configuration blocks must contain a GroupID; this is the first entry in the
configuration block.
7. Configuration blocks may contain one or more TrapOID, VarbindOID pairs.
```

`GroupID` is the first entry in the configuration block. If a GroupID contains a `","` or `">"` these symbols must be preceded by `"/"`. GroupIDs may not contain `"#"`. The `GroupID` does not have to be unique. However, to avoid inconsistent mapping it is recommended that all configuration blocks start with a unique GroupID.

`TrapOID / VarbindOID` are OIDPattern pairs. A `TrapOID` is always followed by a `VarbindOID`. The OIDPattern format follows:

```
1. The pattern must be a Valid OID (Numbers separated by ".")
2. The pattern can optionally contain 1 range (Low#-High#)
3. The pattern can optionally contain 1 wild card "*". A wild card can only occur at the
END of the pattern.
     Example OID Patterns:
          .1.3.6.1.1-6.4.6.*
          .1.3.6.1.5.4.6.*
          .1.3.6.1.5.4.6
```

```
      Invalid OID Patterns:
           .* (Which matches all OIDs is not allowed)
           .1.3.5-4.3.1 (Range must be low to high)
           .1.3.4-5.3.1-3 (Contains 2 ranges)
           .1.3.*.4.1 (wild cards can only occur at the end of the pattern)
```

For generic traps like `linkUp`, you can append the enterprise OID of the vendor to the trap OID for configuration to a specific vendor. Conversely, to configure a generic trap from all vendors you need to append a wild card to the trap OID.

Resolving all traps is not allowed. So the following OIDPattern is ignored: `.*`

EXAMPLES

The following configuration will allow the filter to detect trap storms from CiscoModuleDown traps, on a ModuleIndex basis.

```
< ModuleID=, .1.3.6.1.4.1.9.5.0.4, .1.3.6.1.4.1.9.5.1.3.1.1.1.* >
```

The following configuration will allow the filter to detect trap storms from both CiscoModuleDown and CiscoModuleUp traps. Like the first example, these traps are normalized to the ModuleIndex

```
< ModuleID=, .1.3.6.1.4.1.9.5.0.4, .1.3.6.1.4.1.9.5.1.3.1.1.1.*
.1.3.6.1.4.1.9.5.0.3, .1.3.6.1.4.1.9.5.1.3.1.1.1.*  >
```

The following configuration will allow the filter to detect trap storms from both CiscoModuleDown and CiscoModuleUp traps. This configuration is less verbose than the previous example.

```
< ModuleID=, .1.3.6.1.4.1.9.5.0.3-4, .1.3.6.1.4.1.9.5.1.3.1.1.1.* >
```

This configuration will allow the filter to detect trap storms from both CiscoModuleDown and CiscoModuleUp traps, this configuration will also block all sub CiscoModule Down/Up traps. It is recommended that this pattern be followed. Devices sometimes append additional information on to the trapOID. Without this wild card those traps would not be considered for trap storm analysis.

```
< ModuleID=, .1.3.6.1.4.1.9.5.0.3-4.*, .1.3.6.1.4.1.9.5.1.3.1.1.1.* >
```

The default configuration, which spans multiple lines and has embedded comments, follows. This configuration allows the filter to detect the 2 generic SNMP traps: snmpLinkUp and snmpLinkDown, on an InterfaceIndex basis.

```
<
     InterfaceID=
          #Trap configuration to detect trap storms on link up/down based off of the
source's IfIndex
          .1.3.6.1.6.3.1.1.5.3-4.*, .1.3.6.1.2.1.2.2.1.1.*
>
```

AUTHOR

`hosted-object-trapstorm.conf` was developed by Hewlett-Packard Company.

FILES

`$NnmDataDir/shared/nnm/conf/hosted-object-trapstorm.conf`

SEE ALSO

nnmtrapconfig.ovpl(1M).

Return to Reference Pages Index

Name

UnnumberedNodeGroup.conf — Node group name that is used to identify those nodes unnumbered interfaces should be discovered with

SYNOPSIS

`UnnumberedNodeGroup`

DESCRIPTION

`UnnumberedNodeGroup.conf` is a file used by NNMi to determine whether devices should be evaluated for the existence of unnumbered interfaces, and thus infer potential L2 connection across them. This file must contain the name of a single node group, or it can be the name of a parent node group that represents multiple child node groups containing the device identifiers.

If this file is missing or specified node group name does not exist in the NNMi system, unnumbered interface management will not be enabled. In addition, the administrator must create the `UnnumberedNodeGroup.conf` file. It does not exist by default.

If you modify the `UnnumberedNodeGroup.conf` file while NNMi processes are running, you must restart NNMi.

EXAMPLES

The following is an example of a `UnnumberedNodeGroup.conf` file:

```
Unnumbered Node Group
```

In the example, A node group named `Unnumbered Node Group` is existing in NNMi.

AUTHOR

`UnnumberedNodeGroup.conf` was developed by Hewlett-Packard Company.

FILES

`$NnmDataDir/shared/nnm/conf/disco/UnnumberedNodeGroup.conf`

SEE ALSO

UnnumberedSubnets.conf(4).

Return to Reference Pages Index

Name

UnnumberedSubnets.conf — Contains a list of subnets that should be looked up by the NNMi against routing tables on the devices for unnumbered interface management

SYNOPSIS

`UnnumberedSubnets.conf`

DESCRIPTION

`UnnumberedSubnets.conf` is a file used by NNMi to scope down the routing table query for unnumbered interface management. If you do not create and configure this file, NNMi will do the full MIB-II routing table walk against devices; By using the `UnnumberedSubnets.conf` file, NNMi requests MIB data for only those routes falling in the specified subnet ranges. It is a good practice to use this file and reduce the amount of discovery traffic and performance effect on the devices.

The `UnnumberedSubnets.conf` file can have one or more lines of CIDR subnets. The order of the subnets is not important, and NNMi will sort them out before querying the routing table. The range of subnet is inclusive.

The administrator must create the `UnnumberedSubnets.conf` file. It does not exist by default.

If you modify the `UnnumberedSubnets.conf` file while NNMi processes are running, you must restart NNMi.

EXAMPLES

The following is an example of a `UnnumberedSubnets.conf` file:

```
#This entry filters the following routes: 10.1.0-63.
10.1.5.0/18
#This entry filters the following routes: 15.2.*.*
15.2.126.0/16
#This entry filters the following routes: 192.168.1.0-255
192.168.1.0/24
```

In the example, instead of full routing table walk, NNMi only queries routing tables ipRoutingTable or ipCidrRoutingTable for routes in the specified ranges.

AUTHOR

`UnnumberedSubnets.conf` was developed by Hewlett-Packard Company.

FILES

```
$NnmDataDir/shared/nnm/conf/disco/UnnumberedSubnets.conf
```

SEE ALSO

UnnumberedNodeGroup.conf(4).

Return to Reference Pages Index

Name

nnmincidentcfg.format — file containing incident configurations that can be loaded into the NNMi database. This file format is created by `nnmincidentcfgdump.ovpl` and loaded into the database by `nnmincidentcfgload.ovpl`

SYNOPSIS

`nnmincidentcfg.format`

DESCRIPTION

The `nnmincidentcfg.format` file contains NNMi incident configurations that can be loaded into the NNMi database. This file uses a required set of tags to identify its content.

Each configuration must start with one of the following configuration type tags that identify the five possible incident configuration types.

```
*ConfigurationType=MgmtEventConfig
*ConfigurationType=PairwiseConfig
*ConfigurationType=RemoteNnmEventConfig
*ConfigurationType=SnmpTrapConfig
*ConfigurationType=SyslogMessageConfig
```

When editing the incident configurations, note the following:

```
*The pound sign (#) denote comments.
*All comments must appear before the configuration type tag.
*If a pound sign (#) appears within the configuration data, it is treated as part of the
current tag’s value.
*Comments are not saved in the NNMi database. Therefore, they do not appear in the
output from a subsequent nnmincidentdump.ovpl command.
*All tags that appear after a configuration type tag are considered to be part of the
configuration for that incident configuration type.
*You can modify any tag that begins with a dash (-).
*You cannot modify any tags that begin with an asterisk (*) after they have been
imported for the first time.
*(OPTIONAL) denotes that a tag  is optional.
*Brackets ([]) indicate that the tag value must comply with a specified format or list
of valid values.
*Tags that are annotated with the text “(Direct child tags may occur multiple times)”
are a placeholder for a list of child configuration tags.
*The UUID tag is optional.  UUIDs are used by NNMi as unique database identifiers. Do
not define UUIDs in a configuration file.
*NNMi creates Label tags if you do not provide them.
*Tags that require a Key/Label result in a validation error under the following
circumstances:
*You do not provide either the Key and Label value.
*NNMi is unable to determine the Label from the Key value provided.
```

Note the following exception: NNMi assigns “Customer” as the Author Key and Label value for any incident configuration that is changed.

Before loading an nnmincidentcfg.format file, try the following recommended process:

```
1. Use the nnmincidentcfgdump.ovpl command with the –name option to select an example
incident configuration that is the type of incident you want to edit.
2. Examine the output so that you can identify the tag hierarchy.
3. After you are familiar with the file format, locate the incident configuration type
in the list of examples below, determine the type of change you want to make, and insert
```

```
the information where is it required.
4. Validate your changes using the nnmincidentcfgload.ovpl -validate command.
5. Test your changes by loading the file into the NNMi database using the
nnmincidentcfgload.ovpl -import command.
```

NOTE: The nnmincidentcfgload.ovpl command generates errors for those values that do not match the required format.

FILES

NNMi provides example configuration files and a description of the valid formats in the following directory:

```
Windows: install_dir\examples\nnm\incidentcfg
UNIX: /opt/OV/examples/nnm/incidentcfg
```

EXAMPLES

Create a management event configuration using only the required tags:

```
*ConfigurationType=MgmtEventConfig
        *Name MinimnalistMgmtConfig
        *Oid .1.3.6.1.4.1.11.2.17.19.2.0.9999
        -Author
                -Key com.customer.author
        -Category
                -Key com.hp.nms.incident.category.Fault
        -Family
                -Key com.hp.nms.incident.family.Node
        -MessageFormat Custom message format
        -Severity MINOR
```

Add an enrichment configuration to the management event incident:

```
*ConfigurationType=MgmtEventConfig
        *Name MinimnalistMgmtConfig
        *Oid .1.3.6.1.4.1.11.2.17.19.2.0.9999
        -Author
                -Key com.customer.author
        -Category
                -Key com.hp.nms.incident.category.Fault
        -Family
                -Key com.hp.nms.incident.family.Node
        -MessageFormat Custom message format
    -EnrichConfiguration
        -Enable true
        -Enrichments
            -Enrichment
                -PayloadFilter
                    -Expression ciaName notEquals "varArg"
```

Configure a syslog message incident configuration that has comments before the configuration type tag:

```
#
# Insert comments before the configuration type tag
#
# NNMi does not store comments in the NNMi database
#
# This example includes only the required tabs for the syslog message configuration
#
*ConfigurationType=SyslogMessageConfig
        *Name MinimalistSyslogConfig
        -Author
                -Key com.minimal.customer
                -Label MinimalCustomer
        -Category
```

```
                -Key com.hp.nms.incident.category.Fault
        -Family
                -Key com.hp.nms.incident.family.AggregatePort
        -MessageFormat $.1.3.6.1.4.1.11937.1.54.5: $.1.3.6.1.4.1.11937.1.4
        -Severity CRITICAL
```

PAIRWISE CONFIGURATION FORMAT

The following example contains the valid format for `PairwiseConfig` configuration types.

```
*ConfigurationType=PairwiseConfig (ROOT TAG)
    *Name
    -SetOfPairItems (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -SetOfPairItem (OPTIONAL TAG)
            -FirstInPair
            -FirstParamType
            -SecondInPair
            -SecondParamType
            *UUID (OPTIONAL TAG)
    -Author
        -Key
        -Label (OPTIONAL TAG)
    -DeleteWhenClosed (OPTIONAL TAG)
    -Description (OPTIONAL TAG)
    -Duration
    -Enable (OPTIONAL TAG)
    -FirstIncidentConfigRef
        -Key
        -Type = [MgmtEventConfig, SnmpTrapConfig, SyslogMessageConfig,
RemoteNnmEventConfig]
    -FirstIncidentName
    -FirstIncidentPayloadFilter (OPTIONAL TAG)
        -Expression [Format = Formatted Expression String]
        *UUID (OPTIONAL TAG)
    -SecondIncidentConfigRef
        -Key
        -Type = [MgmtEventConfig, SnmpTrapConfig, SyslogMessageConfig,
RemoteNnmEventConfig]
    -SecondIncidentName
    -SecondIncidentPayloadFilter (OPTIONAL TAG)
        -Expression [Format = Formatted Expression String]
        *UUID (OPTIONAL TAG)
```

MANAGEMENT EVENT CONFIGURATION FORMAT

The following example contains the valid format for `MgmtEventConfig` configuration types.

```
*ConfigurationType=MgmtEventConfig (ROOT TAG)
    *Name
    *Oid
    -Author
        -Key
        -Label (OPTIONAL TAG)
    -Category
        -Key
        -Label (OPTIONAL TAG)
    -Enable (OPTIONAL TAG)
    -ActionConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -Action (OPTIONAL TAG)
                -Command (OPTIONAL TAG)
                -CommandType
                -LifecycleState = [Registered, InProgress, Completed, Closed, Dampened]
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -DampenConfiguration (OPTIONAL TAG)
```

```
        -Enable (OPTIONAL TAG)
        -HourInterval
        -MinuteInterval
        -SecondInterval
        *UUID (OPTIONAL TAG)
        -PayloadFilter (OPTIONAL TAG)
            -Expression [Format = Formatted Expression String]
            *UUID (OPTIONAL TAG)
    -DedupConfiguration (OPTIONAL TAG)
        -ComparisonCriteria
        -CorrelationIncidentConfig (OPTIONAL TAG)
            *Name
        -DedupCount (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        -HourInterval (OPTIONAL TAG)
        -MinuteInterval (OPTIONAL TAG)
        -SecondInterval (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -ComparisonParamList (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -ComparisonParam (OPTIONAL TAG)
                -ParamType (OPTIONAL TAG)
                -ParamValue
                *UUID (OPTIONAL TAG)
    -Description (OPTIONAL TAG)
    -Family
        -Key
        -Label (OPTIONAL TAG)
    -MessageFormat
    -Severity
    -EnrichConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -Enrichment (OPTIONAL TAG)
                -AssignedTo (OPTIONAL TAG)
                -Category (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -Description (OPTIONAL TAG)
                -Family (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -MessageFormat (OPTIONAL TAG)
                -Nature (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
                -Priority (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -EnrichCia (OPTIONAL TAG)
                        -CiaName
                        -EnrichCiaType
                        -Expression [Format = Formatted Expression String]
                        *UUID (OPTIONAL TAG)
                -Severity (OPTIONAL TAG)
    -SuppressConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -PayloadFilter (OPTIONAL TAG)
            -Expression [Format = Formatted Expression String]
            *UUID (OPTIONAL TAG)
    -InterfaceGroups (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -InterfaceGroup (OPTIONAL TAG)
            -Enable
            *UUID (OPTIONAL TAG)
            -DampenConfiguration
                -Enable (OPTIONAL TAG)
                -HourInterval
                -MinuteInterval
                -SecondInterval
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
            -EnrichConfiguration
                -Enable (OPTIONAL TAG)
```

```
                *UUID (OPTIONAL TAG)
                -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Enrichment (OPTIONAL TAG)
                        -AssignedTo (OPTIONAL TAG)
                        -Category (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -Description (OPTIONAL TAG)
                        -Family (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -MessageFormat (OPTIONAL TAG)
                        -Nature (OPTIONAL TAG)
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
                        -Priority (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple
times)
                            -EnrichCia (OPTIONAL TAG)
                                -CiaName
                                -EnrichCiaType
                                -Expression [Format = Formatted Expression String]
                                *UUID (OPTIONAL TAG)
                        -Severity (OPTIONAL TAG)
            *InterfaceGroup
            -Ordering
            -ActionConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Action (OPTIONAL TAG)
                        -Command (OPTIONAL TAG)
                        -CommandType
                        -LifecycleState = [Registered, InProgress, Completed, Closed,
Dampened]
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
            -SuppressConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -NodeGroups (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -NodeGroup (OPTIONAL TAG)
            -Enable (OPTIONAL TAG)
            *UUID (OPTIONAL TAG)
            -DampenConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                -HourInterval
                -MinuteInterval
                -SecondInterval
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
            -EnrichConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Enrichment (OPTIONAL TAG)
                        -AssignedTo (OPTIONAL TAG)
                        -Category (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -Description (OPTIONAL TAG)
                        -Family (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -MessageFormat (OPTIONAL TAG)
                        -Nature (OPTIONAL TAG)
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
```

```
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
                        -Priority (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple
times)
                            -EnrichCia (OPTIONAL TAG)
                                -CiaName
                                -EnrichCiaType
                                -Expression [Format = Formatted Expression String]
                                *UUID (OPTIONAL TAG)
                        -Severity (OPTIONAL TAG)
            -FlowDefinitions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                -FlowDefinition (OPTIONAL TAG)
                    -Enable (OPTIONAL TAG)
                    *FlowDefinition
                    -LifecycleState
                        -Key
                        -Label (OPTIONAL TAG)
                    *UUID (OPTIONAL TAG)
            *NodeGroup
            -Ordering
            -ActionConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Action (OPTIONAL TAG)
                        -Command (OPTIONAL TAG)
                        -CommandType
                        -LifecycleState = [Registered, InProgress, Completed, Closed,
Dampened]
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
            -SuppressConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -RateConfiguration (OPTIONAL TAG)
        -ComparisonCriteria
        -CorrelationIncidentConfig (OPTIONAL TAG)
            *Name
        -Enable (OPTIONAL TAG)
        -HourInterval (OPTIONAL TAG)
        -MinuteInterval (OPTIONAL TAG)
        -RateCount (OPTIONAL TAG)
        -SecondInterval (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -ComparisonParamList (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -ComparisonParam (OPTIONAL TAG)
                -ParamType (OPTIONAL TAG)
                -ParamValue
                *UUID (OPTIONAL TAG)
```

SNMP TRAP CONFIGURATION FORMAT

The following example contains the valid format for `SnmpTrapConfig` configuration types.

```
*ConfigurationType=SnmpTrapConfig (ROOT TAG)
    *Name
    *Oid
    -Author
        -Key
        -Label (OPTIONAL TAG)
    -Category
        -Key
        -Label (OPTIONAL TAG)
    -Enable (OPTIONAL TAG)
    -ActionConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
```

```
          -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
              -Action (OPTIONAL TAG)
                  -Command (OPTIONAL TAG)
                  -CommandType
                  -LifecycleState = [Registered, InProgress, Completed, Closed, Dampened]
                  *UUID (OPTIONAL TAG)
                  -PayloadFilter (OPTIONAL TAG)
                      -Expression [Format = Formatted Expression String]
                      *UUID (OPTIONAL TAG)
      -DampenConfiguration (OPTIONAL TAG)
          -Enable (OPTIONAL TAG)
          -HourInterval
          -MinuteInterval
          -SecondInterval
          *UUID (OPTIONAL TAG)
          -PayloadFilter (OPTIONAL TAG)
              -Expression [Format = Formatted Expression String]
              *UUID (OPTIONAL TAG)
      -DedupConfiguration (OPTIONAL TAG)
          -ComparisonCriteria
          -CorrelationIncidentConfig (OPTIONAL TAG)
              *Name
          -DedupCount (OPTIONAL TAG)
          -Enable (OPTIONAL TAG)
          -HourInterval (OPTIONAL TAG)
          -MinuteInterval (OPTIONAL TAG)
          -SecondInterval (OPTIONAL TAG)
          *UUID (OPTIONAL TAG)
          -ComparisonParamList (OPTIONAL TAG) (Direct child tags may occur multiple times)
              -ComparisonParam (OPTIONAL TAG)
                  -ParamType (OPTIONAL TAG)
                  -ParamValue
                  *UUID (OPTIONAL TAG)
      -Description (OPTIONAL TAG)
      -Family
          -Key
          -Label (OPTIONAL TAG)
      -GeoCentralForwardConfiguration (OPTIONAL TAG)
          -Enable (OPTIONAL TAG)
          *UUID (OPTIONAL TAG)
          -PayloadFilter (OPTIONAL TAG)
              -Expression [Format = Formatted Expression String]
              *UUID (OPTIONAL TAG)
      -MessageFormat
      -Severity
      -EnrichConfiguration (OPTIONAL TAG)
          -Enable (OPTIONAL TAG)
          *UUID (OPTIONAL TAG)
          -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
              -Enrichment (OPTIONAL TAG)
                  -AssignedTo (OPTIONAL TAG)
                  -Category (OPTIONAL TAG)
                      -Key
                      -Label (OPTIONAL TAG)
                  -Description (OPTIONAL TAG)
                  -Family (OPTIONAL TAG)
                      -Key
                      -Label (OPTIONAL TAG)
                  -MessageFormat (OPTIONAL TAG)
                  -Nature (OPTIONAL TAG)
                  *UUID (OPTIONAL TAG)
                  -PayloadFilter (OPTIONAL TAG)
                      -Expression [Format = Formatted Expression String]
                      *UUID (OPTIONAL TAG)
                  -Priority (OPTIONAL TAG)
                      -Key
                      -Label (OPTIONAL TAG)
                  -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple times)
                      -EnrichCia (OPTIONAL TAG)
                          -CiaName
                          -EnrichCiaType
                          -Expression [Format = Formatted Expression String]
                          *UUID (OPTIONAL TAG)
                  -Severity (OPTIONAL TAG)
      -SuppressConfiguration (OPTIONAL TAG)
          -Enable (OPTIONAL TAG)
          *UUID (OPTIONAL TAG)
          -PayloadFilter (OPTIONAL TAG)
              -Expression [Format = Formatted Expression String]
```

```
            *UUID (OPTIONAL TAG)
    -InterfaceGroups (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -InterfaceGroup (OPTIONAL TAG)
            -Enable
            *UUID (OPTIONAL TAG)
            -DampenConfiguration
                -Enable (OPTIONAL TAG)
                -HourInterval
                -MinuteInterval
                -SecondInterval
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
            -EnrichConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Enrichment (OPTIONAL TAG)
                        -AssignedTo (OPTIONAL TAG)
                        -Category (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -Description (OPTIONAL TAG)
                        -Family (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -MessageFormat (OPTIONAL TAG)
                        -Nature (OPTIONAL TAG)
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
                        -Priority (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple
times)
                            -EnrichCia (OPTIONAL TAG)
                                -CiaName
                                -EnrichCiaType
                                -Expression [Format = Formatted Expression String]
                                *UUID (OPTIONAL TAG)
                        -Severity (OPTIONAL TAG)
            *InterfaceGroup
            -Ordering
            -ActionConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Action (OPTIONAL TAG)
                        -Command (OPTIONAL TAG)
                        -CommandType
                        -LifecycleState = [Registered, InProgress, Completed, Closed,
Dampened]
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
            -SuppressConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -NodeGroups (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -NodeGroup (OPTIONAL TAG)
            -Enable (OPTIONAL TAG)
            *UUID (OPTIONAL TAG)
            -DampenConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                -HourInterval
                -MinuteInterval
                -SecondInterval
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
            -EnrichConfiguration (OPTIONAL TAG)
```

```
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Enrichment (OPTIONAL TAG)
                        -AssignedTo (OPTIONAL TAG)
                        -Category (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -Description (OPTIONAL TAG)
                        -Family (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -MessageFormat (OPTIONAL TAG)
                        -Nature (OPTIONAL TAG)
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
                        -Priority (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple
times)
                            -EnrichCia (OPTIONAL TAG)
                                -CiaName
                                -EnrichCiaType
                                -Expression [Format = Formatted Expression String]
                                *UUID (OPTIONAL TAG)
                        -Severity (OPTIONAL TAG)
            -FlowDefinitions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                -FlowDefinition (OPTIONAL TAG)
                    -Enable (OPTIONAL TAG)
                    *FlowDefinition
                    -LifecycleState
                        -Key
                        -Label (OPTIONAL TAG)
                    *UUID (OPTIONAL TAG)
            *NodeGroup
            -Ordering
            -ActionConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Action (OPTIONAL TAG)
                        -Command (OPTIONAL TAG)
                        -CommandType
                        -LifecycleState = [Registered, InProgress, Completed, Closed,
Dampened]
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
            -SuppressConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -RateConfiguration (OPTIONAL TAG)
        -ComparisonCriteria
        -CorrelationIncidentConfig (OPTIONAL TAG)
            *Name
        -Enable (OPTIONAL TAG)
        -HourInterval (OPTIONAL TAG)
        -MinuteInterval (OPTIONAL TAG)
        -RateCount (OPTIONAL TAG)
        -SecondInterval (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -ComparisonParamList (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -ComparisonParam (OPTIONAL TAG)
                -ParamType (OPTIONAL TAG)
                -ParamValue
                *UUID (OPTIONAL TAG)
    -UserRootCause (OPTIONAL TAG)
```

SYSLOG MESSAGE CONFIGURATION FORMAT

The following example contains the valid format for `SyslogMessageConfig` configuration types.

```
*ConfigurationType=SyslogMessageConfig (ROOT TAG)
    *Name
    -Author
        -Key
        -Label (OPTIONAL TAG)
    -Category
        -Key
        -Label (OPTIONAL TAG)
    -Enable (OPTIONAL TAG)
    -ActionConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -Action (OPTIONAL TAG)
                -Command (OPTIONAL TAG)
                -CommandType
                -LifecycleState = [Registered, InProgress, Completed, Closed, Dampened]
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -DampenConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        -HourInterval
        -MinuteInterval
        -SecondInterval
        *UUID (OPTIONAL TAG)
        -PayloadFilter (OPTIONAL TAG)
            -Expression [Format = Formatted Expression String]
            *UUID (OPTIONAL TAG)
    -DedupConfiguration (OPTIONAL TAG)
        -ComparisonCriteria
        -CorrelationIncidentConfig (OPTIONAL TAG)
            *Name
        -DedupCount (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        -HourInterval (OPTIONAL TAG)
        -MinuteInterval (OPTIONAL TAG)
        -SecondInterval (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -ComparisonParamList (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -ComparisonParam (OPTIONAL TAG)
                -ParamType (OPTIONAL TAG)
                -ParamValue
                *UUID (OPTIONAL TAG)
    -Description (OPTIONAL TAG)
    -Family
        -Key
        -Label (OPTIONAL TAG)
    -GeoCentralForwardConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -PayloadFilter (OPTIONAL TAG)
            -Expression [Format = Formatted Expression String]
            *UUID (OPTIONAL TAG)
    -MessageFormat
    -Severity
    -EnrichConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -Enrichment (OPTIONAL TAG)
                -AssignedTo (OPTIONAL TAG)
                -Category (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -Description (OPTIONAL TAG)
                -Family (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -MessageFormat (OPTIONAL TAG)
                -Nature (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
```

```
                -Priority (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -EnrichCia (OPTIONAL TAG)
                        -CiaName
                        -EnrichCiaType
                        -Expression [Format = Formatted Expression String]
                        *UUID (OPTIONAL TAG)
                -Severity (OPTIONAL TAG)
    -SuppressConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -PayloadFilter (OPTIONAL TAG)
            -Expression [Format = Formatted Expression String]
            *UUID (OPTIONAL TAG)
    -InterfaceGroups (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -InterfaceGroup (OPTIONAL TAG)
            -Enable
            *UUID (OPTIONAL TAG)
            -DampenConfiguration
                -Enable (OPTIONAL TAG)
                -HourInterval
                -MinuteInterval
                -SecondInterval
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
            -EnrichConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Enrichment (OPTIONAL TAG)
                        -AssignedTo (OPTIONAL TAG)
                        -Category (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -Description (OPTIONAL TAG)
                        -Family (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -MessageFormat (OPTIONAL TAG)
                        -Nature (OPTIONAL TAG)
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
                        -Priority (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple
times)
                            -EnrichCia (OPTIONAL TAG)
                                -CiaName
                                -EnrichCiaType
                                -Expression [Format = Formatted Expression String]
                                *UUID (OPTIONAL TAG)
                        -Severity (OPTIONAL TAG)
            *InterfaceGroup
            -Ordering
            -ActionConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Action (OPTIONAL TAG)
                        -Command (OPTIONAL TAG)
                        -CommandType
                        -LifecycleState = [Registered, InProgress, Completed, Closed,
Dampened]
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
            -SuppressConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
```

```
                    *UUID (OPTIONAL TAG)
    -NodeGroups (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -NodeGroup (OPTIONAL TAG)
            -Enable (OPTIONAL TAG)
            *UUID (OPTIONAL TAG)
            -DampenConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                -HourInterval
                -MinuteInterval
                -SecondInterval
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
            -EnrichConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Enrichment (OPTIONAL TAG)
                        -AssignedTo (OPTIONAL TAG)
                        -Category (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -Description (OPTIONAL TAG)
                        -Family (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -MessageFormat (OPTIONAL TAG)
                        -Nature (OPTIONAL TAG)
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
                        -Priority (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple
times)
                            -EnrichCia (OPTIONAL TAG)
                                -CiaName
                                -EnrichCiaType
                                -Expression [Format = Formatted Expression String]
                                *UUID (OPTIONAL TAG)
                        -Severity (OPTIONAL TAG)
            -FlowDefinitions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                -FlowDefinition (OPTIONAL TAG)
                    -Enable (OPTIONAL TAG)
                    *FlowDefinition
                    -LifecycleState
                        -Key
                        -Label (OPTIONAL TAG)
                    *UUID (OPTIONAL TAG)
            *NodeGroup
            -Ordering
            -ActionConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Action (OPTIONAL TAG)
                        -Command (OPTIONAL TAG)
                        -CommandType
                        -LifecycleState = [Registered, InProgress, Completed, Closed,
Dampened]
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
            -SuppressConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -RateConfiguration (OPTIONAL TAG)
        -ComparisonCriteria
        -CorrelationIncidentConfig (OPTIONAL TAG)
            *Name
        -Enable (OPTIONAL TAG)
        -HourInterval (OPTIONAL TAG)
```

```
        -MinuteInterval (OPTIONAL TAG)
        -RateCount (OPTIONAL TAG)
        -SecondInterval (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -ComparisonParamList (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -ComparisonParam (OPTIONAL TAG)
                -ParamType (OPTIONAL TAG)
                -ParamValue
                *UUID (OPTIONAL TAG)
    -UserRootCause (OPTIONAL TAG)
```

REMOTE NNM EVENT CONFIGURATION FORMAT

The following example contains the valid format for `RemoteNnmEventConfig` configuration types.

```
*ConfigurationType=RemoteNnmEventConfig (ROOT TAG)
    *Name
    *Oid
    -Author
        -Key
        -Label (OPTIONAL TAG)
    -Category
        -Key
        -Label (OPTIONAL TAG)
    -Enable (OPTIONAL TAG)
    -ActionConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -Action (OPTIONAL TAG)
                -Command (OPTIONAL TAG)
                -CommandType
                -LifecycleState = [Registered, InProgress, Completed, Closed, Dampened]
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -DampenConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        -HourInterval
        -MinuteInterval
        -SecondInterval
        *UUID (OPTIONAL TAG)
        -PayloadFilter (OPTIONAL TAG)
            -Expression [Format = Formatted Expression String]
            *UUID (OPTIONAL TAG)
    -DedupConfiguration (OPTIONAL TAG)
        -ComparisonCriteria
        -CorrelationIncidentConfig (OPTIONAL TAG)
            *Name
        -DedupCount (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        -HourInterval (OPTIONAL TAG)
        -MinuteInterval (OPTIONAL TAG)
        -SecondInterval (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -ComparisonParamList (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -ComparisonParam (OPTIONAL TAG)
                -ParamType (OPTIONAL TAG)
                -ParamValue
                *UUID (OPTIONAL TAG)
    -Description (OPTIONAL TAG)
    -Family
        -Key
        -Label (OPTIONAL TAG)
    -GeoCentralForwardConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -PayloadFilter (OPTIONAL TAG)
            -Expression [Format = Formatted Expression String]
            *UUID (OPTIONAL TAG)
    -MessageFormat
    -Severity
    -EnrichConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
```

```
        *UUID (OPTIONAL TAG)
        -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -Enrichment (OPTIONAL TAG)
                -AssignedTo (OPTIONAL TAG)
                -Category (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -Description (OPTIONAL TAG)
                -Family (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -MessageFormat (OPTIONAL TAG)
                -Nature (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
                -Priority (OPTIONAL TAG)
                    -Key
                    -Label (OPTIONAL TAG)
                -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -EnrichCia (OPTIONAL TAG)
                        -CiaName
                        -EnrichCiaType
                        -Expression [Format = Formatted Expression String]
                        *UUID (OPTIONAL TAG)
                -Severity (OPTIONAL TAG)
    -SuppressConfiguration (OPTIONAL TAG)
        -Enable (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -PayloadFilter (OPTIONAL TAG)
            -Expression [Format = Formatted Expression String]
            *UUID (OPTIONAL TAG)
    -InterfaceGroups (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -InterfaceGroup (OPTIONAL TAG)
            -Enable
            *UUID (OPTIONAL TAG)
            -DampenConfiguration
                -Enable (OPTIONAL TAG)
                -HourInterval
                -MinuteInterval
                -SecondInterval
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
            -EnrichConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Enrichment (OPTIONAL TAG)
                        -AssignedTo (OPTIONAL TAG)
                        -Category (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -Description (OPTIONAL TAG)
                        -Family (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -MessageFormat (OPTIONAL TAG)
                        -Nature (OPTIONAL TAG)
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
                        -Priority (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple
 times)
                            -EnrichCia (OPTIONAL TAG)
                                -CiaName
                                -EnrichCiaType
                                -Expression [Format = Formatted Expression String]
                                *UUID (OPTIONAL TAG)
                        -Severity (OPTIONAL TAG)
            *InterfaceGroup
            -Ordering
            -ActionConfiguration
```

```
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Action (OPTIONAL TAG)
                        -Command (OPTIONAL TAG)
                        -CommandType
                        -LifecycleState = [Registered, InProgress, Completed, Closed,
Dampened]
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
            -SuppressConfiguration
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
    -NodeGroups (OPTIONAL TAG) (Direct child tags may occur multiple times)
        -NodeGroup (OPTIONAL TAG)
            -Enable (OPTIONAL TAG)
            *UUID (OPTIONAL TAG)
            -DampenConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                -HourInterval
                -MinuteInterval
                -SecondInterval
                *UUID (OPTIONAL TAG)
                -PayloadFilter (OPTIONAL TAG)
                    -Expression [Format = Formatted Expression String]
                    *UUID (OPTIONAL TAG)
            -EnrichConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Enrichments (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Enrichment (OPTIONAL TAG)
                        -AssignedTo (OPTIONAL TAG)
                        -Category (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -Description (OPTIONAL TAG)
                        -Family (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -MessageFormat (OPTIONAL TAG)
                        -Nature (OPTIONAL TAG)
                        *UUID (OPTIONAL TAG)
                        -PayloadFilter (OPTIONAL TAG)
                            -Expression [Format = Formatted Expression String]
                            *UUID (OPTIONAL TAG)
                        -Priority (OPTIONAL TAG)
                            -Key
                            -Label (OPTIONAL TAG)
                        -EnrichCias (OPTIONAL TAG) (Direct child tags may occur multiple
times)
                            -EnrichCia (OPTIONAL TAG)
                                -CiaName
                                -EnrichCiaType
                                -Expression [Format = Formatted Expression String]
                                *UUID (OPTIONAL TAG)
                        -Severity (OPTIONAL TAG)
            -FlowDefinitions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                -FlowDefinition (OPTIONAL TAG)
                    -Enable (OPTIONAL TAG)
                    *FlowDefinition
                    -LifecycleState
                        -Key
                        -Label (OPTIONAL TAG)
                    *UUID (OPTIONAL TAG)
            *NodeGroup
            -Ordering
            -ActionConfiguration (OPTIONAL TAG)
                -Enable (OPTIONAL TAG)
                *UUID (OPTIONAL TAG)
                -Actions (OPTIONAL TAG) (Direct child tags may occur multiple times)
                    -Action (OPTIONAL TAG)
                        -Command (OPTIONAL TAG)
                        -CommandType
                        -LifecycleState = [Registered, InProgress, Completed, Closed,
```

```
 Dampened]
                          *UUID (OPTIONAL TAG)
                          -PayloadFilter (OPTIONAL TAG)
                              -Expression [Format = Formatted Expression String]
                              *UUID (OPTIONAL TAG)
             -SuppressConfiguration (OPTIONAL TAG)
                 -Enable (OPTIONAL TAG)
                 *UUID (OPTIONAL TAG)
                 -PayloadFilter (OPTIONAL TAG)
                     -Expression [Format = Formatted Expression String]
                     *UUID (OPTIONAL TAG)
    -RateConfiguration (OPTIONAL TAG)
        -ComparisonCriteria
        -CorrelationIncidentConfig (OPTIONAL TAG)
            *Name
        -Enable (OPTIONAL TAG)
        -HourInterval (OPTIONAL TAG)
        -MinuteInterval (OPTIONAL TAG)
        -RateCount (OPTIONAL TAG)
        -SecondInterval (OPTIONAL TAG)
        *UUID (OPTIONAL TAG)
        -ComparisonParamList (OPTIONAL TAG) (Direct child tags may occur multiple times)
            -ComparisonParam (OPTIONAL TAG)
                -ParamType (OPTIONAL TAG)
                -ParamValue
                *UUID (OPTIONAL TAG)
    -UserRootCause (OPTIONAL TAG)
```

AUTHOR

`nnmincidentcfg.format` was developed by Hewlett-Packard Company.

SEE ALSO

nnmincidentcfgload.ovpl (1M) .

nnmincidentcfgdump.ovpl (1M) .

Return to Reference Pages Index

Name

nnmsnmpbulk.ovpl — Queries a node for information using an SNMPv2c GetBulk request.

SYNOPSIS

`nnmsnmpbulk.ovpl -u user_name -p passwd` [*`options`*] *`node object-id`* [*`,object-id`*]...

`options:` [-d] [-v *`version`*] [-c *`community`*] [-port *`port(default:161)`*] [-t *`timeout(default:5000)`*] [-r *`retries(default:1)`*] [-T] [-n *`non-repeaters`*] [-m *`max-repetitions`*] [-pp *`Proxy Port`*] [-pa *`Proxy Address`*] [-a *`Authentication Protocol`*] [-A *`Authentication Pass phrase`*] [-x *`Privacy Protocol`*] [-X *`Privacy Passphrase`*] [-N *`Context Name`*] [-v3u *`SNMPv3 user name`*] [-jndiHost *`hostname`*] [-jndiPort *`port`* Default is 1099]

DESCRIPTION

The `nnmsnmpbulk.ovpl` script uses the SNMPv2c/v3 GetBulk request to retrieve information from an SNMP agent. The SNMP GetBulk request minimizes the number of protocol exchanges required to retrieve a large amount of information. This increases performance, as the `nnmsnmpbulk.ovpl` script uses fewer SNMP requests to retrieve management information from the remote node.

If the `node` is an SNMPv1-only agent, the `nnmsnmpbulk.ovpl` script automatically downgrades the GetBulk request to an SNMPv1-supported GetNext request.

`node` can be a system with an IP address that supports SNMP. You can supply IP nodes to the `nnmsnmpbulk.ovpl` script using either a node's IP address or its hostname.

Supply command arguments for one or more `OIDs` in dotted decimal format or as a mnemonic name. Before attempting to supply a mnemonic name, load the MIB that defines the OID by using the `nnmloadmib.ovpl` script.

Only users who belong to System, Administrator or Web Service Client roles can run the `nnmsnmpbulk.ovpl` script. Users who are in Level1, Level2 or Guest roles cannot run the `nnmsnmpbulk.ovpl` script.

Parameters

`-d`

Dumps all SNMP packets to standard output in a hexadecimal and decoded ASN.1 format.

`-v` *`version`*

Requests a specific version of SNMP be used to communicate with the remote node. Valid choices for *`version`* are `1`, `2`, `2c`, or `3`.

If you do not specify the version, the `nnmsnmpbulk.ovpl` script uses `2c` as the default for nodes not in

the topology.

`-c` *`community`*

Specifies the community string to use for authentication on the remote node.

Note: If the community string contains characters the shell interferes with, use one or more escape symbols or quotation marks as required.

`-port` *`port`*

Specifies the port to use to communicate with the remote node.

`-t` *`timeout`*

Specifies a timeout period, in milliseconds, for communication with the remote node.

`-r` *`retries`*

Specifies the number of retries to use for communication with the remote node.

`-T`

Prints the OID in dotted decimal format and the MIB variable value with no textual conventions applied.

`-n`

`non-repeaters` specifies the number of variables for which a single lexicographic successor is to be returned. This value represents the number of non-repeating varbinds (values) you want the `nnmsnmpbulk.ovpl` script to retrieve.

`-m`

Max-Repetitions specifies the number of lexicographic successors to be returned for the remaining variables. This value represents the number of rows to retrieve for a repeating varbind (value). The repeating varbind is present in each table row.

`-pp` *`Proxy Port`*

Specifies the Proxy Port to use in communication with the node

`-pa` *`Proxy Address`*

Specifies the Proxy IP Address to use to communicate with the node.

`-a` *`Authentication Protocol`*

SNMPv3 Authentication Protocol (MD5|SHA)

`-A` *`Authentication Passphrase`*

SNMPv3 Authentication Passphrase

`-x` *`Privacy Protocol`*

SNMPv3 Privacy Protocol (DES|3DES|AES|AES192|AES256)

`-X` *`Privacy Passphrase`*

SNMPv3 Privacy Passphrase

`-N` *`context`*

SNMPv3 Context Name (for example, vlan1)

`-v3u` *`SNMPv3 user name`*

SNMPv3 security name (for example, testV3user)

`-u <`*`username`*`>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <`*`password`*`>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost<`*`jndiHost`*`>`

The hostname of the server running the jboss application server. If you do not specify a hostname, the `nnmcommload.ovpl` script uses `localhost` as the default value.

`-jndiPort<`*`jndiPort`*`>`

The jboss application server port. If you do not specify this port, the `nnmcommload.ovpl` script uses 1099 as the default value.

If the `nnmsnmpbulk.ovpl` script does not receive a response, it uses a linear backoff algorithm based on the *`timeout`* and *`retries`* arguments to resend the SNMP request. For example, if the *`timeout`* argument is `2000` (two seconds) and the *`retries`* argument is `3`, the initial request would time out after two seconds, the first retry would time out after four seconds, the second retry would time out after six seconds, and the last retry would time out after eight seconds. The `nnmsnmpbulk.ovpl` script requires additional time to resolve the configuration.

EXAMPLES

The following command retrieves all information under the system MIB OID from node `testnode`:

`nnmsnmpbulk.ovpl –c community` *`testnode`* `.1.3.6.1.2.1.1.0`

AUTHOR

`nnmsnmpbulk.ovpl` was developed by Hewlett-Packard Company.

FILES

The environment variable below represents a universal path that is established according to your shell and platform requirements:

Windows: `%NNM_BIN%\nnmsnmpbulk.ovpl`

UNIX: `$NNM_BIN/nnmsnmpbulk.ovpl`

SEE ALSO

nnmloadmib.ovpl(1M), nnmsnmpnotify.ovpl(1M), nnmsnmpset.ovpl(1M), nnmsnmpwalk.ovpl(1M).

RFC 1155, 1157, 1212: SNMP Version 1.

RFC 1901 - 1908, 2576, 2578, 3416 - 3418: SNMP Version 2.

RFC 3411 - 3415: SNMP Version 3.

EXTERNAL INFLUENCES

Environmental Variables

`$LANG` determines the language in which messages appear. If `$LANG` is not specified or is set to an empty string, a default of `C` is used instead of `$LANG`. If any internationalization variable contains an invalid setting, the `nnmsnmpbulk.ovpl` script functions as if all internationalization variables are set to `C`.

International Code Set Support

Supports single-byte and multiple-byte character code sets.

NOTE: SNMP MIB values of the type `DISPLAY STRING` are restricted to NVT-ASCII.

Return to Reference Pages Index

Name

nnmsnmpwalk.ovpl — Query a node using SNMP GET or GETNEXT requests

SYNOPSIS

`nnmsnmpwalk.ovpl -u user_name -p passwd` [*options*] *node object-id*

`nnmsnmpget.ovpl -u user_name -p passwd` [*options*] *node object-id* [*,object-id*]...

`nnmsnmpnext.ovpl -u user_name -p passwd` [*options*] *node object-id* [*,object-id*]...

options: [-d] [-v *version*] [-c *community*] [-port *port(default:161)*] [-t *timeout(default:5000)*] [-r *retries(default:1)*] [-T] [-pp *Proxy Port*] [-pa *Proxy Address*] [-a *Authentication Protocol*] [-A *Authentication Pass phrase*] [-x *Privacy Protocol*] [-X *Privacy Passphrase*] [-N *Context Name*] [-v3u *SNMPv3 user name*] [-jndiHost *hostname*] [-jndiPort *port* Default is 1099]

DESCRIPTION

The `nnmsnmpwalk.ovpl` script sends repeated SNMP GETNEXT requests to retrieve values for all instances of MIB objects registered on node `node`. The `nnmsnmpwalk.ovpl` script determines whether to use SNMP Version 1 or Community-based SNMP Version 2 (SNMPv2c) or version 3, based on the value supplied for the `-v` option and the type of remote node. If you do not specify a variable, the `nnmsnmpwalk.ovpl` script retrieves all values beneath `object.iso.org`. If you do supply a variable, the variable's value determines the starting point in the object identifier space that is searched. For example, the `nnmsnmpwalk.ovpl` script retrieves the entire system group if you supply .1.3.6.1.2.1.1.1. as a variable value. The `nnmsnmpwalk.ovpl` script terminates when all object information beneath the specified variable has been returned.

The `nnmsnmpget.ovpl` script uses the SNMP Get request to query `node` for information.

Normally an SNMP instance number needs to be appended, such as using .0 in .1.3.6.1.2.1.1.1.0 to get the `system.sysDescr.0 value`).

The `nnmsnmpnext.ovpl` script performs the same action as the `nnmsnmpwalk.ovpl` script, except that the `nnmsnmpnext.ovpl` script only returns a single value.

`node` can be an IP-addressable system that supports SNMP, or a target name for which an SNMP proxy configuration is defined. You can identify IP nodes by Internet address or hostname.

You might supply one or more variables as arguments to any of these scripts. Each variable is an object identifier in dotted decimal format or mnemonic name. If you plan to specify the variables by mnemonic name, use the `nnmloadmib.ovpl` script to load the MIB that defines the object identifier before using this method.

If you attempt to search beyond the end of the remote node's MIB with either the `nnmsnmpwalk.ovpl` or `nnmsnmpnext.ovpl` scripts, the scripts display an `End of MIB` message.

Only users who belong to System, Administrator or Web Service Client roles can run these scripts. Users who are in Level1, Level2 or Guest roles cannot run these commands.

Options

`-d`

Dumps all SNMP packets to standard output in a hexadecimal and decoded ASN.1 format.

`-v` *`version`*

Requests the script to use a specific version of SNMP to communicate with the remote node. Valid choices for *`version`* are `1`, `2c`, or `3`.

If you do not specify the version, the script uses `2c` as the default for nodes not in the topology.

`-c` *`community`*

Specifies the community string to use for authentication on the remote node.

Note: If the community string contains characters the shell interferes with, use one or more escape symbols or quotation marks as required.

`-port` *`port`*

Specifies the port to use in communication with the remote node.

`-t` *`timeout`*

Specifies a timeout period, in milliseconds, for communication with the remote node.

`-r` *`retries`*

Specifies the number of retries to use for communication with the remote node.

`-T`

Prints the OID in dotted decimal format and the MIB variable value with no textual conventions applied.

`-pp` *`Proxy Port`*

Specifies the Proxy Port to use in communication with the node

`-pa` *`Proxy Address`*

Specifies the Proxy IP Address to use in communication with the node

`-a` *`Authentication Protocol`*

SNMPv3 Authentication Protocol (MD5|SHA)

`-A` *`Authentication Passphrase`*

SNMPv3 Authentication Passphrase

`-x` *`Privacy Protocol`*

SNMPv3 Privacy Protocol (DES|3DES|AES|AES192|AES256)

`-X` *`Privacy Passphrase`*

SNMPv3 Privacy Passphrase

`-N` *`context`*

SNMPv3 Context Name (for example, vlan1)

`-v3u` *`SNMPv3 user name`*

SNMPv3 security name (for example, testV3user)

`-u` <*`username`*>

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p` <*`password`*>

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost`<*`jndiHost`*>

The hostname of the server running the jboss application server. If you do not specify a hostname, the `nnmcommload.ovpl` script uses `localhost` as the default value.

`-jndiPort`<*`jndiPort`*>

The jboss application server port. If you do not specify this port, the `nnmcommload.ovpl` script uses 1099 as the default value.

If the `nnmsnmpget.ovpl, nnmsnmpnext.ovpl, or nnmsnmpnnmsnmpwalk.ovpl` scripts do not receive a response, a linear backoff algorithm based on the *`timeout`* and *`retries`* arguments is used to resend the SNMP request. For example, if the *`timeout`* argument is `2000` (two seconds) and the *`retries`* argument is `3`, the initial request would time out after two seconds, the first retry would time out after four seconds, the second retry would time out after six seconds, and the last retry would time out after eight seconds. The `nnmsnmpget.ovpl, nnmsnmpnext.opvl, and nnmsnmpwalk.ovpl` scripts require additional time to resolve the configuration.

EXAMPLES

The following script usage requests the system subtree for the node *`testnode`*:

`nnmsnmpwalk.ovpl` *`testnode`* `system`

Output for the above command typically resembles the following:

```
nnmsnmpwalk.ovpl -c community 10.97.1.7 .1.3.6.1.2.1.1
.1.3.6.1.2.1.1.1.0: ASN_OCTET_STR = Ethernet Switch 470-24T-PWR
```

```
.1.3.6.1.2.1.1.2.0: ASN_OBJECT_ID = .1.3.6.1.4.1.45.3.63.1
.1.3.6.1.2.1.1.3.0: ASN_TIMETICKS = 63050579
.1.3.6.1.2.1.1.4.0: ASN_OCTET_STR = Bob Jones 933-558-3453
.1.3.6.1.2.1.1.5.0: ASN_OCTET_STR = wr3-2-front-storage-n91-60-2
.1.3.6.1.2.1.1.6.0: ASN_OCTET_STR = Woods Run 3 2nd floor
.1.3.6.1.2.1.1.7.0: ASN_INTEGER = 3
```

AUTHOR

`nnmsnmpwalk.ovpl`, `nnmsnmpget.ovpl`, and `nnmsnmpnext.ovpl` were developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_BIN%\nnmsnmpwalk.ovpl`

Windows: `%NNM_BIN%\nnmsnmpget.ovpl`

Windows: `%NNM_BIN%\nnmsnmpnext.ovpl`

UNIX: `$NNM_BIN/nnmsnmpwalk.ovpl`

UNIX: `$NNM_BIN/nnmsnmpget.ovpl`

UNIX: `$NNM_BIN/nnmsnmpnext.ovpl`

For information about universal paths for your platform and shell, see the nnm.envvars(1) reference page.

SEE ALSO

nnmsnmpset.ovpl(1M), nnmsnmpbulk.ovpl(1M), nnmsnmpnotify.ovpl(1M).

RFC 1155, 1157, 1212: SNMP Version 1.

RFC 1901 - 1908, 2576, 2578, 3416 - 3418: SNMP Version 2.

RFC 3411 - 3415: SNMP Version 3.

EXTERNAL INFLUENCES

Environmental Variables

`$LANG` determines the language in which messages appear. If `$LANG` is not specified or is set to an empty string, a default of `C` is used instead of `$LANG`. If any internationalization variable contains an invalid setting, `nnmsnmpget.ovpl` behaves as if all internationalization variables are set to `C`.

International Code Set Support

Supports single-byte and multiple-byte character code sets.

NOTE: SNMP MIB values of the type `DISPLAY STRING` are restricted to NVT-ASCII.

Return to Reference Pages Index

Name

nnmsnmpset.ovpl — issue an SNMP set request

SYNOPSIS

`nnmsnmpset.ovpl -u user_name -p passwd` [*`options`*] *`node object-id asnType value`* [*`object-id asnType value`*]...

`options:` [-d] [-v *`version`*] [-c *`write community`*] [-port *`port(default:161)`*] [-t *`timeout(default:5000)`*] [-r *`retries(default:1)`*] [-T] [-pp *`Proxy Port`*] [-pa *`Proxy Address`*] [-a *`Authentication Protocol`*] [-A *`Authentication Pass phrase`*] [-x *`Privacy Protocol`*] [-X *`Privacy Passphrase`*] [-N *`Context Name`*] [-v3u *`SNMPv3 user name`*] [-jndiHost *`hostname`*] [-jndiPort *`port`* Default is 1099]

DESCRIPTION

The `nnmsnmpset.ovpl` script sends an SNMP set request to alter MIB objects on the remote *`node`*.

`node` can be an IP-addressable system that supports SNMP, or a target name for which an SNMP proxy configuration is defined. You can identify IP nodes by an Internet address or by a hostname.

The `nnmsnmpset.ovpl` script passes data to the remote node as a triple of *`object-id`*,*`asnType`*,*`value`*. Supply one or more triples to the `nnmsnmpset.ovpl` script as command-line arguments.

Each *`object-id`* is an object instance identifier in dotted decimal format (for example, `.1.3.6.1.4.1.11.2.17.2.1.0` or mnemonic string format (for example, `openViewSourceId.0`).

Each *`asnType`* must be one of the following asnTypes:

integer

integer32

unsigned32

octetstring

octetstringhex

octetstringoctal

octetstringascii

objectidentifier

null

ipaddress

counter

counter32

counter64 (for SNMPv2c or v3 capable remote nodes)

gauge

gauge32

timeticks

opaque

opaquehex

opaqueoctal

opaqueascii

For a complete description of each *asnType*, refer to *RFC 1155* and *RFC 1902*.

The *value* parameter must be valid for the asnType specified. When using a asnType that requires a hexadecimal or octal value, you must fully define each byte of the value. For example, if you specify `fff` (or `17377`), it is missing a byte, and will not work. Use `0fff` (or `017377`) instead. The `nnmsnmpset.ovpl` script ignores the *null* asnType. You must specify a *value* on the command line. If you try to use a *null* type, the `nnmsnmpset.ovpl` script ignores the `null` type when it creates the request. *value* must not be larger than 512 bytes.

Only users who belong to System, Administrator or Web Service Client roles can run the `nnmsnmpset.ovpl` script. Users who are in Level1, Level2 or Guest roles cannot run the `nnmsnmpset.ovpl` script.

Parameters

`-d`

Dumps all SNMP packets to standard output in a hexadecimal and decoded ASN.1 format.

`-v` *version*

Requests the `nnmsnmpset.ovpl` script to use a specific version of SNMP to communicate with the remote node. Valid choices for *version* are `1`, `2c`, or `3`.

If you do not specify the version, the `nnmsnmpset.ovpl` script uses `2c` as the default for nodes not in the topology.

`-c` *write community*

Specifies the write community string to use for authentication on the remote node.

Note: If the community string contains characters the shell interferes with, use one or more escape symbols or quotation marks as required.

`-port` *port*

Specifies the port to use in communication with the remote node.

`-t` *`timeout`*

Specifies a timeout period, in milliseconds, for communication with the remote node.

`-r` *`retries`*

Specifies the number of retries to use for communication with the remote node.

`-T`

Prints the OID in dotted decimal format and the MIB variable value with no textual conventions applied.

`-pp` *`Proxy Port`*

Specifies the Proxy Port to use in communication with the node

`-pa` *`Proxy Address`*

Specifies the Proxy IP Address to use in communication with the node

`-a` *`Authentication Protocol`*

SNMPv3 Authentication Protocol (MD5|SHA)

`-A` *`Authentication Passphrase`*

SNMPv3 Authentication Passphrase

`-x` *`Privacy Protocol`*

SNMPv3 Privacy Protocol (DES|3DES|AES|AES192|AES256)

`-X` *`Privacy Passphrase`*

SNMPv3 Privacy Passphrase

`-N` *`context`*

SNMPv3 Context Name (for example, vlan1)

`-v3u` *`SNMPv3 user name`*

SNMPv3 security name (for example, testV3user)

`-u` *`<username>`*

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p` *`<password>`*

Supply the NNMi administrator password to run the script. This script requires the NNMi

administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-jndiHost<`*`jndiHost`*`>` (optional)

The hostname of the server running the jboss application server. If you do not specify a hostname, the `nnmcommload.ovpl` script uses `localhost` as the default value.

`-jndiPort<`*`jndiPort`*`>` (optional)

The jboss application server port. If you do not specify this port, the `nnmcommload.ovpl` script uses 1099 as the default value.

If the `nnmsnmpset.ovpl` script does not receive a response, it uses a linear backoff algorithm based on the *`timeout`* and *`retries`* arguments to resend the SNMP request. For example, if the *`timeout`* argument is `2000` (two seconds) and the *`retries`* argument is `3`, the initial request would time out after two seconds, the first retry would time out after four seconds, the second retry would time out after six seconds, and the last retry would time out after eight seconds. The `nnmsnmpset.ovpl` script requires additional time to resolve the configuration.

EXAMPLES

The following command sets the system contact to Bob Jones for the node testnode:

`nnmsnmpset.ovpl -c writeCommunity` *`testnode`* `system.sysContact.0 octetstring "Bob Jones"`

The output for this command is as follows:

```
system.sysContact.0 OCTET STRING-(ascii): Bob Jones
```

AUTHOR

`nnmsnmpset.ovpl` was developed by Hewlett-Packard Company.

FILES

See the `nnm.envvars` reference page (and the UNIX manpage) for information about using environment variables for the following files:

`install_dir``\bin\nnmsnmpset.ovpl`

`install_dir``\doc\rfc*.txt`

SEE ALSO

nnmsnmpwalk.ovpl(1M), nnmsnmpbulk.ovpl(1M), nnmsnmpnotify.ovpl(1M).

RFC 1155, 1157, 1212: SNMP Version 1.

RFC 1901 - 1908, 2576, 2578, 3416 - 3418: SNMP Version 2.

RFC 3411 - 3415: SNMP Version 3.

EXTERNAL INFLUENCES

Environmental Variables

`$LANG` determines the language in which messages appear. If `$LANG` is not specified or is set to an empty string, a default of `C` is used instead of `$LANG`. If any internationalization variable contains an invalid setting, `nnmsnmpset.ovpl` behaves as if all internationalization variables are set to `C`.

International Code Set Support

Supports single-byte and multiple-byte character code sets.

NOTE: SNMP MIB values of the asnType `octetstringascii` are restricted to NVT-ASCII.

Return to Reference Pages Index

Name

nnmsso.ovpl — reload single sign-on configuration.

SYNOPSIS

`nnmsso.ovpl` -reload

DESCRIPTION

`nnmsso.ovpl` is a script that enables you to reload the single sign-on (SSO) configuration from the `nms-ui.properties` file without restarting ovjboss.

Note: Other applications integrating with NNMi might use a different configuration file for enabling and configuring single sign-on. See the integrating product's documentation for instructions about enabling single sign-on and setting the user interface initialization string (initString) parameter that needs to be configured.

Parameters

The `nnmsso.ovpl` script supports the following options:

`-reload`

Reloads the SSO configuration.

EXAMPLES

`nnmsso.ovpl -reload`

Reads modifications made to the `nms-ui.properties` file. These changes could include enabling SSO, disabling SSO, or changing the initString.

AUTHOR

`nnmsso.ovpl` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_PROPS%\nms-ui.properties`

UNIX: `$NNM_PROPS/nms-ui.properties`

SEE ALSO

nnmofficialfqdn.ovpl(1M), nnmsetofficialfqdn.ovpl(1M), nnmldap.ovpl(1M).

Return to Reference Pages Index

Name

nnmstatuspoll.ovpl — update the status for a node using the State Poller

SYNOPSIS

`nnmstatuspoll.ovpl` [–node <*`nodename`*|*`IP Address`*> [-tenant *`tenant name`*] [-t *`timeout in secs`*] [-v]]

DESCRIPTION

The `nnmstatuspoll.ovpl` script enables you to dynamically poll a device that is being monitored. This results in a refresh of key collected state values. When all of the information for the state demand poll has been collected and displayed, the `nnmstatuspoll.ovpl` script informs you that the task that you requested is complete.

Parameters

`-node <`*`nodename`*`|`*`IP Address`*`>`

Target node name or IP address.

`-tenant <`*`tenant name`*`>`

Tenant the given node is paired with. This option is useful when node names and IP addresses are non-unique in the topology such as can occur in overlapping address domain environments.

`-t <`*`timeout in secs`*`>`

Client waits till given timeout in sec.

`-v`

Displays the detailed verbose log message on console.

`-jndiHost <`*`hostname`*`>`

Jboss server host. Default is `localhost`.

`-jndiPort <`*`port`*`>`

Jboss server port. Default is `1099`.

`-u <`*`username`*`>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <`*`password`*`>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-help`

Prints the usage statement.

RETURN VALUE

`nnmstatuspoll.ovpl` returns the appropriate output shown in the above Parameters section.

When using `-v` option, you see the information in the following columns:

Column 1: Indicates which protocol is used to collect the data.

Column 2: Indicates which device name was polled.

Column 3: Indicates which MIB instance was polled.

Column 4: Indicates the result of the poll.

Column 5: Indicates a mapped value, if it exists.

AUTHOR

`nnmstatuspoll.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnm.properties(4)

Return to Reference Pages Index

Name

nnmtopodump.ovpl — Displays the contents of the NNMi topology database

SYNOPSIS

`nnmtopodump.ovpl` [-h] -u <*`username`*> -p <*`password`*> -type <*`type`*> [-legacy <*`format`*>] [-filter <*`filter`*>]

DESCRIPTION

`nnmtopodump.ovpl` displays the contents of the topology database. By default, NNMi displays the output in xml format unless you specify the -legacy option.

Parameters

The `nnmtopodump.ovpl` script supports the following options:

`-h`

Displays the usage statement.

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-http.host <host>`

The server host; default is localhost.

`-http.port <port>`

The server port; default is 8004.

`-type <type>`

The type of the objects available to be printed. Use one of the following types: node|interface|incident|ip|subnet|rrp|vlan|nodeSensor|interfaceAggregation|card|l2connection

`-legacy [short|long]`

If you use the legacy option, NNMi displays the data in text output. If you do not specify this option,

then NNMi displays the output in xml form. Use the legacy option with the following type values only: node, interface, ip, l2connection and interfaceAggregation. For -legacy short, this is only valid for the type values node, interface, and l2connection.

`-filter <filter>`

Filters the output by property. The `nnmtopodump.ovpl` script supports the following filters:

node - node.name | node.shortname | node.id | node.uuid | node.status | node.snmpaddress | node.managementMode | node.deviceCategory | node.deviceDescription | node.deviceFamily | node.deviceVendor

interface - node.name | node.shortname | node.id | node.snmpaddress | node.deviceCategory | node.deviceDescription | node.deviceFamily | node.deviceVendor | node.managementMode | interface.ifType | interface.id | interface.uuid | interface.managementMode | interface.managementState

ip - interface.id | node.id | ip.value | ip.id

vlan - node.name | node.id | vlan.id | vlan.name | vlan.value

nodeSensor - node.name | node.hostname | node.id | nodeSensor.id | nodeSensor.name | nodeSensor.type

card - node.name | node.hostname | node.id | card.id |card.name

l2connection - connection.name | connection.id | connection.uuid

interfaceAggregation - master.id | master.uuid | master.index | master.alias | slave.id | slave.uuid | slave.index | slave.alias

EXAMPLES

```
nnmtopodump.ovpl -u username -p password -type node
```

Displays all of the nodes in the topology database in xml format. (You must provide an NNMi administrator username and password.)

```
nnmtopodump.ovpl -u username -p password -legacy long -type node
```

To display the nodes in text format, you must use the legacy option. When you use the legacy option with -type node, NNMi displays the nodes and their interfaces.

Equivalent command in NNM 6.x/7.x: `ovtopodump -l`

```
nnmtopodump.ovpl -u username -p password -type node -filter node.name=foo.hp.com
```

Display information about node foo.hp.com in xml format.

```
nnmtopodump.ovpl -u username -p password -legacy long -type node -filter
node.name=foo.hp.com
```

Display information about node foo.hp.com in text format. When you use the legacy option with -type node, it displays the nodes with the interfaces attached to the node.

```
ovtopodump -lr foo.hp.com
```

Equivalent command in NNM 6.x/7.x:

```
nnmtopodump.ovpl -u username -p password -legacy long -type node -filter node.id=2345
```

Display information about the node having nodeid as 2345 in text format. NNMi also displays all interfaces of the node.

Equivalent command in NNM 6.x/7.x: `ovtopodump -lr 2345`

```
nnmtopodump.ovpl -u username -p password -type interface -filter
interface.managementState=MANAGED
```

Display information about all interfaces that are managed. Filter values can be MANAGED, INHERITED, NOTMANAGED, and OUTOFSERVICE.

AUTHOR

`nnmtopodump.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnmnodedelete.ovpl(1M), nnm.properties(4).

Return to Reference Pages Index

Name

nnmtrapconfig.ovpl — Configure HP NNM Trap service

SYNOPSIS

`nnmtrapconfig.ovpl` -u <*user*> -p <*password*> [-showProp] [-start] [-stop] [-readFilter] [-dumpBlockList] [-resetBlockCache]

`nnmtrapconfig.ovpl` -setProp -u <*user*> -p <*password*> [trapInterface <*ip_addr*>] [unsetTrapInterface] [trapPort <*port*>] [recvSocketBufSize <*size*>] [loopbackAddrOverride <*ip_addr*>] [resetLoopbackAddrOverride] [blockTraps] [unblockTraps] [thresholdRate <*rate*>] [rearmRate <*rate*>] [overallThresholdRate <*rate*>] [overallRearmRate <*rate*>] [windowSize <*time*>] [updateSourcesPeriod <*time*>] [notifySourcesPeriod <*time*>] [minTrapCount <*count*>] [numSources <*count*>] [databaseQSize <*count*>] [pipelineQSize <*count*>] [databaseFileSize <*size*>] [databaseFileCount <*count*>] [hostedObjectTrapstorm <*boolean*>] [hostedObjectThreshold <*rate*>] [trapLoggingMode <*log mode*>] [trapLoggingCompression <*boolean*>] [trapLoggingMaxFileSize <*size*>] [trapLoggingRollAttempts <*count*>] [trapLoggingTaskInterval <*time*>] [trapLoggingBatchSize <*size*>] [-persist]

DESCRIPTION

`nnmtrapconfig.ovpl` can be used to display or modify the current properties of the Trap Service. In addition it can be used to start or stop the Trap Service. It also provides the following filter related functionalities: read filter configuration files, print out the current filter configuration and blocking caches or reset the blocking caches

Parameters

`nnmtrapconfig.ovpl` supports the following parameters:

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-showProp`

Show the properties associated with Trap Service and their current values.

`-start`

Start the Trap Service.

`-stop`

Stop the Trap Service.

`-readFilter`

Cause Trap Service to read the filter configuration files.

`-dumpBlockList`

Print out the filter configuration and the blocking caches that the Trap Service creates.

`-resetBlockCache`

Reset the blocking caches in the Trap Service.

To set values for one or more properties, run `nnmtrapconfig.ovpl` with the following options

`-u` *`user`*

The name of a user with system or administrator roles.

`-p` *`password`*

The password for the specified user.

`-setProp`

Set values for properties associated with Trap Service.

`trapInterface` *`ip address`*

Set IP address on which the Trap Service listens for traps. By default it listens on all interfaces.

`unsetTrapInterface`

Set Trap Service to listen on all interfaces.

`trapPort` *`port`*

Set the port on which the Trap Service listens for traps. By default this is 162.

`recvSocketBufSize` *`size`*

Set the size of socket buffer, in kilobytes, on which Trap Service listens for traps.

`loopbackAddrOverride` *`ip address`*

Sets the IP Address to replace the source address of a trap if the source address is a loopback address before forwarding the trap.

`resetLoopbackAddrOverride`

Resets the loopback override address. When this options is executed, the user supplied loopback override address will be removed. In this case, NNM chooses one of the server's addresses as the loopback override address.

`blockTraps`

Block traps based on filter and threshold configurations.

`unblockTraps`

Do not block traps.

`thresholdRate` *`rate`*

Set the rate in traps/sec at which trap sources or trap oids are blocked.

`rearmRate` *`rate`*

Set the rate in traps/sec at which blocked trap sources or trap oids are unblocked. This rate should be less than or equal to the `thresholdRate`.

`overallThresholdRate` *`rate`*

Set the rate in traps/sec at which all incoming traps are blocked.

`overallRearmRate` *`rate`*

Set the rate in traps/sec at which all incoming traps are unblocked. This rate should be less than or equal to the `overallThresholdRate`

`windowSize` *`time`*

Set monitoring window size in seconds. This determines the window size in which trap sources are monitored. This means that whenever this time expires the counters keeping track of incoming traps are reset and a new window is started.

`updateSourcesPeriod` *`time`*

Set the time period in seconds after which the list of blocked traps and sources are updated.

`notifySourcesPeriod` *`time`*

Set the time period in seconds after which the list of blocked traps and sources are reported in the trapanalytics log file under the `NNM_LOG` directory. This is also the time period after which a trap storm incident is generated, in case a trap storm occurs and the `overallThresholdRate` is violated.

`minTrapCount` *`count`*

Set the minimum number of traps to be received from a source before it is considered for blocking. Also the minimum number of the same trap received before that trap is considered for blocking.

`numSources` *`count`*

Set the number of nosiest sources to monitor at any given time. These will be the sources that are sending the most traps as well as the traps that occur most frequently. This information is reported in the trapanalytics log file in the `NNM_LOG` directory.

`databaseQSize` *`count`*

Set the maximum number of traps that can be held in the queue that writes traps to the database.

`pipelineQSize` *`count`*

Set the maximum number of traps that can be held in the queues for each stage of the trap pipeline.

`databaseFileSize` *`size`*

Set the maximum size, in MB, for one file in the trap database. When the file size reaches this value a rollover happens.

`databaseFileCount` *`count`*

Set the maximum number of files in the trap database.

`hostedObjectTrapstorm` *`enabled`*

Enable or disable hosted object trap storm detection and suppression.

`hostedObjectThreshold` *`rate`*

If a hosted object's trap rate is greater than this threshold in seconds, its traps will be suppressed.

`trapLoggingMode` *`log mode`*

The mode that the trap logger will operate with, valid values are: OFF, CSV, TXT, BOTH. Default value: CSV

```
OFF: Turns off all trap logging
CSV: Traps will be logged in a CSV format
TXT: Traps will be logged in a txt format similiar to trapd.log
BOTH: Traps will be logged in both formats
```

`trapLoggingCompression` *`boolean`*

If true, traps will be logged in a gz compression, Default: false

`trapLoggingMaxFileSize` *`size`*

Maximum size in MB that the trap log files will grow to before being rolled archived to a .old file. Only 1 .old file is retained for each log format. Defaul: 5MB

`trapLoggingRollAttempts` *`count`*

Maximum number of attempts the trap logger will try to roll a log file internally. Default: 1

`trapLoggingTaskInterval` *`time`*

Time in seconds the trap logger will wait before writing traps to the filesystem. Defult: 6 seconds

`trapLoggingBatchSize` *`size`*

Maximum number of traps that will be written to the filesystem during each interval. Default: 256

`-persist`

Persist the current properties so that on future restarts these values will be used.

EXAMPLES

Show the properties associated with the Trap Service and their values:

```
nnmtrapconfig.ovpl -u user -p pass -showProp
```

Start the Trap Service:

```
nnmtrapconfig.ovpl -u user -p pass -start
```

Set the trap port to 1162:

```
nnmtrapconfig.ovpl -u user -p pass -setProp trapPort 1162
```

Enable the hosted object trap storm detection and suppression stage, with a per device traps/second threshold of 50. This command will also persist the current values for future invocations of the Trap Service.

```
nnmtrapconfig.ovpl -u user -p pass -setProp hostedObjectTrapstorm true
hostedObjectThreshold 50 -persist
```

Log recieved traps using both formats, with a maximum file size of 32MB, using a task interval of 30 seconds and a batch size of 1024. Also persist the current values for future invocations of Trap Service.

```
nnmtrapconfig.ovpl -u user -p pass -setProp trapLoggingMode BOTH trapLoggingMaxFileSize
32 trapLoggingTaskInterval 30 trapLoggingBatchSize 1024 -persist
```

Enable blocking and also persist the current values for future invocations of Trap Service

```
nnmtrapconfig.ovpl -u user -p pass -setProp blockTraps -persist
```

Persist the current values for future invocations of Trap Service

```
nnmtrapconfig.ovpl -u user -p pass -setProp -persist
```

FILES

The following file stores properties for NNM Trap Service:

Windows:*data_dir*`\shared\nnm\conf\nnmtrapserver.properties`

UNIX:`/var/opt/OV/shared/nnm/conf/nnmtrapserver.properties`

The blocking filters can be configured in the following file:

Windows:*data_dir*`\shared\nnm\conf\nnmtrapd.conf`

UNIX:`/var/opt/OV/shared/nnm/conf/nnmtrapd.conf`

The hosted object trap storm detection and suppression filter can be configured in the following file:

Windows:*data_dir*`\shared\nnm\conf\hosted-object-trapstorm.conf`

UNIX:`/var/opt/OV/shared/nnm/conf/hosted-object-trapstorm.conf`

SEE ALSO

hosted-object-trapstorm.conf(4).

AUTHOR

`nnmtrapconfig.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmtrapdload.ovpl — Merge traps/events from a trapd.conf file with the current database

SYNOPSIS

`nnmtrapdload.ovpl` [[[-loadTrapd <*`trapd_file`*> [-authorLabel <*`author_label`*> -authorKey <*`author_key`*>]] -verbose <*`true`*|*`false`*> -skipExisting <*`true`*|*`false`*> -disableAllTraps <*`true`*|*`false`*> [-u <*`user`*>] [-p <*`password`*>]]

DESCRIPTION

Parses and merges events/traps from an existing trapd.conf file into the database.

`nnmtrapdload.ovpl` should only be used for importing snmp trapd configurations, and NOT 6.x/7.x legacy ov event trap definitions.

If the provided category is not supported it is by default mapped to Status

If the provided severity is not supported it is by default mapped to Normal

The trap/event parser only resolves the tokens EVENT, FORMAT, SDESC, and EDESC, it does not resolve the tokens; EXEC, FORWARD, DISPLAY, or NODES

Parameters

`nnmtrapdload.ovpl` supports the following parameters:

`-u <username>`

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p <password>`

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-loadTrapd` *`trapd_file`*

File location for trapd.conf

`-authorLabel` *`author_label`*

Author label for storing incident configs in the database.

`-authorKey` *`author_key`*

Author key for storing incident configs in the database.

`-verbose` *`true|false`*

If true logs each trap/event with verbose output; default is false

`-skipExisting` *`true|false`*

If true skips over traps/events that already exist in the database (updates/merges are not performed); default is false

`-disableAllTraps` *`true|false`*

If specified all traps should be loaded as disabled; default is false

EXAMPLES

To Load trapd.conf file with verbose output

```
nnmtrapdload.ovpl -loadTrapd /tmp/conf/trapd.conf -verbose true -u user -p password
```

To load trapd.conf file while skipping existing entries

```
nnmtrapdload.ovpl -loadTrapd /tmp/conf/trapd.conf -skipExisting true -u user -p password
```

To load trapd.conf file with all events disabled

```
nnmtrapdload.ovpl -loadTrapd /tmp/conf/trapd.conf -disableAllTraps true -u user -p
password
```

To load trapd.conf file with a supplied author

```
nnmtrapdload.ovpl -loadTrapd /tmp/conf/trapd.conf -authorLabel trapd_parser -authorKey
com.hp.ov.nnm.parser -u user -p password
```

AUTHOR

`nnmtrapdload.ovpl` was developed by Hewlett-Packard Company.

Return to Reference Pages Index

Name

nnmtrimincidents.ovpl — delete and (optionally) archive incidents

SYNOPSIS

`nnmtrimincidents.ovpl` [[[-age *age* -incr *incr*] | -date *date* | -trimOldest *numberToTrim*] [-nature *nature*] [-lifecycle *lifecycleState*] [-severity *severity*] [-origin *origin*] [-name *name*] [-family *family*] [-sysobjectid *sysobjectid*] [-path *path*] [-archiveOnly] [-trimOnly] [-trimAndArchive] [-batch *batchSize*] [-u *username*] [-p *password*] [-quiet]]

DESCRIPTION

`nnmtrimincidents.ovpl` is used to delete incidents from the incident table. The deleted incidents are (optionally) saved in a compressed archive file:

data_dir`\tmp\incidentArchive.ISO 8601 Date.Time Ms.txt.gz`

The default behavior is to delete incidents without archiving.

ARCHIVE-FORMAT

Incidents will be archived using a `csv` format. The column names are ordered as follows:
`TimeStamp(LastOccurance),Name,SourceNodeName,SourceObjectName,SysObjectID(ALWAYS_EMPTY),FormattedMessage,LifeCycleState,Severity,Priority,AssignedTo,JournalNotes,Category,Family,Nature,Origin,IncidentNotes,DuplicateCount,FirstOccuranceTime,OriginOccuranceTime,PayLoad,ElementOID`

```
TimeStamp        - Last time this incident occurred, as a human readable time string.
Name             - Incident Name
SourceNodeName   - Source Node Short Name
SourceObjectName - Sorce Object Name
SysObjectID      - Always empty this field has been kept for backwards compatibility
FormattedMessage - Formatted string describing the incident
LifeCycleState   - Incident lifecycle state
Severity         - Incident severity
Priority         - Incident priority localized label
AssignedTo       - Account/Person holding this incident
JournalNotes     - Incident journal notes
Category         - Incident category localized label
Family           - Incident family localized label
Nature           - Incident nature
Origin           - Incident origin
IncidentNotes    - Incident notes
DuplicateCount   - # of times the incident has occured in the system
FirstOccuranceTime  - In the case of duped incidents this is the first one
OriginOccuranceTime - Timestamp for when the trap/syslog arrived into the system (if any)
PayLoad: All CIAS, "Name[Type]=Value"  seperated by | (In the case of multiple CIA)
       Name      - Name of CIA (50 chars max)
       Type      - Value type of the CIA
       Value     - Value of the CIA (2000 chars max)
ElementOID       - OID for the incident if it exists
```

Parameters

`-age` *age*

Specifies the age of incidents to trim. You should use this option in conjunction with the *incr* option. If specified, the value for *age* must be greater than `0`.

`-incr` *increment*

Specifies the increment for *age* option. Supported increments include `days`, `weeks`, and `months`.

`-trimOldest` *numberToTrim*

Specifies a number of incidents to delete from the database. The oldest *numberToTrim* incidents will be selected from all incidents in the database based on the options specified.

`-date` *date*

Specifies the date from which older incidents are trimmed. The date is specified in ISO 8601 standard format: *yyyy-mm-ddThh:mm:ss[+ or -]hh:mm*.

`-archiveOnly`

Creates an archive file. Does not trim incidents. This option requires you to specify the *age* or *date* option.

`-trimOnly`

Trims incidents without archiving the deleted incidents. This is the default behavior.

`-trimAndArchive`

Trims incidents with archiving the deleted incidents.

`-batch` *batchSize*

Specifies the batch size when trimming incidents. If specified, the value for *batch* must be greater than `0` and less than or equal to `1000`.

`-path` *path*

Specifies the archive file name with a complete path. This path overrides the default archive file:

data_dir`\tmp\incidentArchive.txt.gz`

`-jndiHost` *hostname*

Server `jndi` host. Default is `localhost`.

`-jndiPort` *port*

Server `jndi` port. Default is `1099`.

`-lifecycle` *`lifecycle`*

Specifies the lifecycle state of incidents matching the *`age|date`* to trim.

Example *`lifecycle`* states include:

`Registered`

`InProgress`

`Completed`

`Closed`

`-severity` *`severity`*

Specifies the severity of incidents matching the *`age|date`* to trim.

Example *`severity`* states include:

`Critical`

`Major`

`Minor`

`Warning`

`Normal`

`-name` *`name`*

Specifies the name of incidents matching the *`age|date`* to trim.

`-family` *`family`*

Optionally specify the family of incidents matching `age|date` to trim.

`-sysobjectid` *`sysobjectid`*

Optionally specify the device system object id of incidents matching `age|date` to trim.

`-nature` *`nature`*

Specifies the nature of incidents matching the *`age|date`* to trim.

Examples of *`nature`* include:

`RootCause`

`SecondaryRootCause`

`Symptom`

`ServiceImpact`

`StreamCorrelation`

`None`

`Info`

`Dedup_Stream_Correlation`

`Rate_Stream_Correlation`

`-origin` *`origin`*

Specifies the origin of incidents matching the *`age|date`* to trim.

Example *`origin`* states include:

`ManagementSoftware`

`ManuallyCreated`

`RemotelyGenerated`

`SnmpTrap`

`Syslog`

`Other`

`-u` *`<username>`*

Supply the NNMi administrator username to run the script. This script requires the NNMi administrator username unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-p` *`<password>`*

Supply the NNMi administrator password to run the script. This script requires the NNMi administrator password unless you use an `nnm.properties` file. See the `nnm.properties.4` reference page for more information.

`-quiet` *`quiet`*

Specifies non-prompt mode.>

EXAMPLES

Trim incidents older than 6 days:

```
nnmtrimincidents.ovpl -age 6 -incr days
```

Trim incidents older than 6 weeks with a nature of Symptom:

```
nnmtrimincidents.ovpl -age 6 -incr weeks -nature Symptom
```

Trim incidents older than 6 months with a lifecycle state of Closed:

```
nnmtrimincidents.ovpl -age 6 -incr months -lifecycle Closed
```

Trim incidents older than 6 months with a severity of Normal:

```
nnmtrimincidents.ovpl -age 6 -incr months -severity Normal
```

Trim incidents older than the specified date:

```
nnmtrimincidents.ovpl -date 2007-07-16T19:20:30
```

Trim incidents older than the specified date with a nature of Symptom:

```
nnmtrimincidents.ovpl -date 2007-07-16T19:20:30+01:00 -nature Symptom
```

Trim incidents older than the specified date with a lifecycle state of Closed:

```
nnmtrimincidents.ovpl -date 2007-07-16T19:20:30-01:00 -lifecycle Closed
```

Trim incidents older than 6 days using the specified filename for the archive:

```
nnmtrimincidents.ovpl -age 6 -incr days -path "C:\BkupDir\saveIncidents.gz"
```

Trim incidents older than 6 days (without archiving):

```
nnmtrimincidents.ovpl -age 6 -incr days
```

Trim the oldest 10,000 SNMP traps with Symptom nature:

```
nnmtrimincidents.ovpl -trimOldest 10000 -nature Symptom -origin SnmpTrap
```

Trim incidents older than 6 days (without archiving):

```
nnmtrimincidents.ovpl -trimOnly -age 6 -incr days
```

Archive incidents older than 6 days (without trimming):

```
nnmtrimincidents.ovpl -archiveOnly -age 6 -incr days
```

Trim and archive incidents older than 6 days:

```
nnmtrimincidents.ovpl -trimAndArchive -age 6 -incr days
```

AUTHOR

`nnmtrimincidents.ovpl` was developed by Hewlett-Packard Company.

SEE ALSO

nnm.properties(4)

Return to Reference Pages Index

Name

nnmwhat — get specific NNM information from executables and libraries

SYNOPSIS

```
nnmwhat FILE
```

DESCRIPTION

The `nnmwhat` command is the equivalent command to the `what` command available on most Unix systems. This command is used to pull select string information from files such that specific information can be returned. This includes the copyright, operating system, patch level and product level. This is useful for the patching process and Support when determining the versions of files that are on the system.

Parameters

Provide a file name to execute the command against. If no file is provided then STDIN is used.

EXAMPLES

`nnmwhat ovaddobj.exe` will return the information from the ovaddobj.exe command.

Return to Reference Pages Index

Name

ovaddobj — object registration utility

SYNOPSIS

`ovaddobj` [*`lrf-file`*]

DESCRIPTION

`ovaddobj` is used to register object managers (i.e. agents) with the HP process management proces ovspmd(1M).

Parameters

`lrf-file`

Specifies a Local Registration File (LRF), which must contain information about a single agent and the objects it manages.

Note

You must specify all objects managed by the agent in the same LRF. Running `ovaddobj` against an LRF containing additional objects managed by a previously registered object manager does *not add* those objects. Instead, it *replaces* the previously registered objects with the new objects.

EXAMPLES

`ovaddobj` *`mylrf`*

This registers the agent and all the objects described in the LRF *`mylrf`* into the NNM startup file.

AUTHOR

`ovaddobj` was developed by Hewlett-Packard Company.

FILES

See the `nnm.envvars` reference page (and the UNIX manpage) for information about using environment variables for the following files:

`install_dir``/bin/ovaddobj`

SEE ALSO

ovstatus(1), ovstart(1M), ovstop(1M), ovdelobj(1M), ovspmd(1M), nnmcluster(1).

Return to Reference Pages Index

Name

ovdelobj — object deregistration utility

SYNOPSIS

`ovdelobj` [*`lrf-file`*]

DESCRIPTION

`ovdelobj` is used to deregister the information for object managers (i.e. agents) from the HP process management process ovspmd(1M).

Parameters

`lrf-file`

Specifies a Local Registration File (LRF), which contains information about a single agent and the objects it manages.

EXAMPLES

```
ovdelobj mylrf
```

This deregisters the agent and all the objects described in the LRF `mylrf`.

AUTHOR

`ovdelobj` was developed by Hewlett-Packard Company.

FILES

See the `nnm.envvars` reference page (and the UNIX manpage) for information about using environment variables for the following file:

`install_dir``/bin/ovdelobj`

SEE ALSO

ovstatus(1), ovstart(1M), ovstop(1M), ovaddobj(1M), ovspmd(1M), nnmcluster(1).

Return to Reference Pages Index

Name

ovjboss — wrapper around the jboss Application Server

SYNOPSIS

`ovjboss`

DESCRIPTION

`ovjboss` is a service component that falls under the management of `ovspmd`. It uses properties files (`$NNM_DATA/shared/nnm/conf/props/nms-support.properties` `$NNM_DATA/shared/nnm/conf/props/nms-jboss.properties`, and `$NNM_DATA/shared/nnm/conf/props/ovjboss.jvmargs`) to pass arguments to the jboss application server. Each file contains documentation on how to change the settings it controls.

This command should be never be executed directly but it falls under the management of `ovspmd`. It will be started when running `ovstart` or `ovstart -c ovjboss`. To stop it either call `ovstop` or `ovstop -c ovjboss`. To see the status of internal services it monitors, call `ovstatus -v ovjboss`.

If there are problems starting ovbjoss, one can look at the `ovjboss.log` and `jbossServer.log` log files and see if that might contain information to help solve the problem:

You must be logged on as `administrator` (Windows) or `root` (UNIX) user to run this command.

EXAMPLES

To start *`NNM`* including *`ovjboss`* run the following command:

`$InstallDir/bin/ovstart`

To only start *`ovjboss`* run the following command:

`$InstallDir/bin/ovstart -c` *`ovjboss`*

To find the status of services started by *`ovjboss`* run the following command:

`$InstallDir/bin/ovstatus -v` *`ovjboss`*

AUTHOR

`ovjboss` was developed by Hewlett-Packard Company.

FILES

`$NNM_DATA``/shared/nnm/conf/props/nms-jboss.properties`

Parameter file used by services started inside `ovjboss`.

$NNM_DATA`/shared/nnm/conf/props/nms-support.properties`

Parameter file used by services started inside `ovjboss`.

$NNM_DATA`/shared/nnm/conf/props/ovjboss.jvmargs`

Parameters passed to the JVM that jboss runs in

$NNM_DATA`/nnm/conf/nms-local.properties`

Local configuration file, including Ports configuration

$NNM_DATA`/log/nnm/jbossServer.log`

Log file containing exceptions (if any)

$NNM_DATA`/log/nnm/ovjboss.log`

Log file containing stderr messages

SEE ALSO

ovspmd(1)

nms-local.properties(4)

Return to Reference Pages Index

Name

ovserror — Reports the most recently generated errors from the `ovspmd` process. The `ovserror` process is used internally by other processes, and should never be invoked by the user.

SYNOPSIS

```
ovserror
```

DESCRIPTION

`ovserror` reports the most recently generated errors from the `ovspmd` process. It takes no parameters.

RETURN VALUE

`ovserror` reports the most recently generated errors from the ovspmd process.

AUTHOR

`ovserror` was developed by Hewlett-Packard Company.

SEE ALSO

ovstatus(1), ovstart(1M), ovstop(1M), ovspmd(1M).

Return to Reference Pages Index

Name

ovspmd — NNM process management service

SYNOPSIS

`ovspmd` [[install] [start] [stop] [remove] [-W] [-d] [-V] [-f *`startup_file`*]]

DESCRIPTION

`ovspmd` manages the service processes that are part of NNM. It starts, stops, and reports status on these processes in response to requests from `ovstart`, `ovstop`, and `ovstatus`. `ovspmd` is normally started automatically by `ovstart`. On Windows, `ovspmd` is registered as a service. `ovspmd` registers under the service name HP OpenView Process Manager.

`ovstart` sends a request to `ovspmd` to start the object manager programs specified in the NNM startup file (SUF), by default `ovsuf`. NNM-managed processes are configured in a local registration file (LRF), and added to the SUF by `ovaddobj`. If you call `ovstart` with no arguments, `ovspmd` starts all managed processes configured to be started automatically (that is, with the initial start flag `OVs_YES_START` in the LRF).

`ovstop` sends a request to `ovspmd` to stop configured managed processes. If you call `ovstop` with no arguments, `ovspmd` stops all currently running managed processes, and then exits.

`ovstatus` sends a request to `ovspmd` to report the current running status of configured managed processes.

Managed processes are started by `ovspmd` as services (that is, in the background, with `stdin`, `stdout`, and `stderr` ignored).

Each managed process can be configured with a dependency list (that is, a list of other processes that must already be running before the process can be started successfully). `ovspmd` does not start a managed process until all the processes on which it depends have already initialized successfully. On startup, `ovspmd` verifies that no LRF-specified dependencies form a cycle. (An example of a cycle is `A -> B -> C -> A`.) These dependencies determine a relative sequencing for starting, as well as a reverse order for stopping.

`ovspmd` has a mechanism to automatically restart processes that fail unexpectedly. This process entails adding a retry count for the daemon processes as listed in the `$NNM_DATA/shared/nnm/conf/ovspmd.restart.properties file`. By default, the number of retries is 3. When a process dies unexpectedly, this count is decremented by one until it reaches zero. At that point, the process will not be automatically restarted. Attempting to start the process with `ovstart` will reset the retry count and start the process again. If the process has been running for two hours, then the process resets its retry counter. Removing entries will cause `ovspmd` not to do restarts. This is also true if the retry count is 0.

`ovspmd` distinguishes between three classes of object managers:

`OVs_WELL_BEHAVED`

> A well-behaved process uses the OVsPMD API (see OVsPMD_API(3)) to communicate with `ovspmd`. It sends `ovspmd` status information about successful and unsuccessful initialization, normal termination and abnormal termination, if configured to do so. `ovspmd` considers a well-behaved process to have

initialized successfully only when it explicitly reports that it has done so. A well-behaved process also exits when it receives the command `OVS_CMD_EXIT` from `ovspmd`.

The status information passed by the managed process to `ovspmd` is forwarded to `ovstart`, `ovstop`, or `ovstatus`, if currently running. The last message received from each managed process is saved, and then forwarded, on request, to `ovstatus`. The messages received from well-behaved processes are also logged to the application event log (which can be examined with the Event Viewer).

`OVs_NON_WELL_BEHAVED`

`ovspmd` can also manage object managers that do not use the `OVsPMD` API (non-well-behaved processes) only if they do *not* go into the background of their own accord (see `OVs_DAEMON` below). Because a non-well-behaved process returns no status messages, `ovspmd` considers such a process to have initialized successfully if it is not exited within the LRF-specified timeout interval.

Non-well-behaved processes are terminated with Terminal Process if they do not exit within the configured timeout.

`OVs_DAEMON`

Managed processes that go into the background cannot be managed with a communication channel or with signals. `ovspmd` can start such a process, but it cannot stop or report meaningful status about the process because it does not have a communication channel or a process ID for it.

Parameters

`install`

Install `ovspmd` as a service.

`start`

Start the `ovspmd` service.

`stop`

Stop the `ovspmd` service.

`remove`

Remove the `ovspmd` service.

`-W`

Do not start managed processes when `ovspmd` starts. Wait for `ovstart` to request it.

`-d`

Used for debugging. When used, `ovspmd` does *not* become a service.

`-V`

Run in very verbose mode. In this mode, `ovspmd` outputs very detailed information about the configuration of the managed processes. This is far too much information for ordinary use.

`-f` *`startup_file`*

Read *startup_file* as the startup file (SUF) instead of the default. Note that *startup_file* must be an absolute path.

Application Authorization

ovspmd governs the management of NNM services. It uses the ovspmd.auth file to control which hosts, users, and applications can start and stop the NNM services. The ovspmd.auth file is located in *data_dir*\conf\.

ovspmd searches the entries in the ovspmd.auth file from beginning to end. As soon as it finds an entry that either explicitly allows or denies the access under consideration, it stops looking. Therefore, more specific entries should precede more general entries.

The file contains lines specifying the authorized hosts, users, and applications. Each line lists a single host, user, and application list authorized to connect to ovspmd. The format of each line of the file is:

#comment

hostname [*username* [*appname1 appname2 appname3 ...*]]

The pound sign (#) and anything following it is a comment, which is ignored. Blank lines are also ignored.

username and *appname* are optional. If no application is present, the line permits (or denies) access by any application. If no username is present, the line permits (or denies) access by any user running any application.

If *hostname* is a plus sign (+), the line refers to access from any host. If *username* is a plus sign (+), the line refers to access by any user. If a hostname is preceded by a minus sign (-), the line explicitly denies all access from that host. (Any username or application names that also happen to appear on the line are ignored.) If a username is preceded by a minus sign (-), the line explicitly denies any access by that user from the specified host. (Any application names that also happen to appear on the line are ignored.)

If any applications are listed, the line permits access only to the applications listed (by the specified user from the specified host). Note that the application names listed in the authorization file must match the registered name of the application, except that white space in the registered application name must be replaced with underscores.

The ovspmd.auth file created at installation contains more examples of the file format, and some examples are also included in the EXAMPLES section.

DIAGNOSTICS

ovspmd issues error messages about configuration errors and system call failures. These messages are intended to be self-explanatory. If it currently has an open communication channel with ovstart, ovstop, or ovstatus, ovspmd forwards these error messages through the communication channel to be output by the program.

ovspmd can process multiple requests (start, stop, or status) at a time. Additional requests are queued by type until the current request completes.

In addition, ovspmd logs processing, configuration, and system errors using nettl in the OVS subsystem at the ERROR level. Messages indicating normal events, such as successful initialization, are logged at the

INFORMATIVE level. Messages indicating initialization failure or abnormal termination are logged at the WARNING level.

EXAMPLES

The following is an example of the contents of the ovspmd.auth file:

```
# Normally, you should authorize any application
# run by any user on the same host on which ovspmd is running.
# To do so, use a single line listing the
# name of the host on which this file is located
# (for example, "thishost"):

thishost

# Similarly, if you are running Management
# Consoles, you should authorize any application
# run by any user on all the client hosts and on
# the server host. For example, if your server
# system named "bigsystem" has one client named
# "hohum", list each of them on a separate line in
# this file on bigsystem:

bigsystem
hohum

# It is possible to permit specific users to run
# specific applications from a remote system. The
# following line permits the user "shem" from host
# "blimp" to run the applications "Toaster Manager"
# and "Blender". Note that, because the application's
# registered name "Toaster Manager" contains white
# space, you must replace the whitespace with the
# underscore character in the authorization file:

shem blimp Toaster_Manager Blender

# It is not possible to exclude specific applications,
# except by explicitly permitting all non-excluded
# applications.

# The following line denies access by the user "fred"
# from any host:

+ -fred

# The following line denies any application access
# from the host "badguy":

-badguy
```

AUTHOR

ovspmd was developed by Hewlett-Packard Company.

FILES

See the nnm.envvars reference page (and the UNIX manpage) for information about using environment variables for the following files:

install_dir\bin\ovspmd

install_dir\conf\ovsuf

See $NNM_DATA/shared/nnm/conf/ovspmd.restart.properties for restart property configuration.

EXTERNAL INFLUENCES

Environmental Variables

$LANG provides a default value if the internationalization variables, LC_ALL, LC_CTYPE, and LC_MESSAGES are unset, null, or invalid.

If $LANG is unset, null, or invalid, the default value of C (or English_UnitedStates.1252 on Windows) is used.

LC_ALL (or $LANG) determines the locale of all other processes started by ovspmd.

LC_CTYPE determines the interpretation of text as single-byte characters, multiple-byte characters, or both; the classification of characters as printable; and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the language in which messages are displayed.

All other environment variables are inherited from the shell executing ovspmd (or the initial ovstart that starts ovspmd). ovspmd and all service processes share this same environment. As a result, ovspmd must be stopped and restarted for any environment changes to take effect (see ovstart(1M)).

SEE ALSO

ovstatus(1), ovstart(1M), ovstop(1M), ovaddobj(1M), ovdelobj(1M), nnmcluster(1).

Return to Reference Pages Index

Name

ovstart — start NNM managed processes

SYNOPSIS

`ovstart` [[-c] [-d] [-o *`ovspmd_path`*] [-v] [--][*`ovspmd_options...`*][*`managed_process_names`*...]]

DESCRIPTION

`ovstart` starts NNM managed processes. If called with one or more *`managed_process_name`* arguments, it starts the designated managed process after first starting any other managed processes on which it depends. If called with no arguments, it starts all the managed processes that are configured to start by default.

`ovstart` does not exit until all the managed processes it has tried to start have either responded or timed out (failed to respond within the LRF-specified timeout interval). By default, it produces no output unless a managed process fails. When you execute it from the command line, it is advisable to use the `-c` or `-v` option to track the progress of the operation. Running `ovstart` again after the successful completion of a previous attempt to ovstart is completely harmless.

`ovstart` sends a start request (`OVS_REQ_START`) to the process management service, `ovspmd`. If `ovspmd` is not already running, `ovstart` starts it first.

`ovstart` must be run by the administrator or super-user.

The managed processes are configured by `ovaddobj` from information in local registration files (see lrf(4)). A managed process is named by the first field in the LRF describing it.

If `ovstart` is used on a node configured for NNM clustering (see nnmcluster(1M)) then the behavior of ovstart is different than described above. Specifically, ovstart behaves exactly like the `"nnmcluster -daemon"` command.

In a NNM cluster environment `ovstart` returns immediately (after launching the NNM cluster in the background). Instead, the `nnmcluster` command will determine if/when to start the other NNM processes. Please monitor `ovstatus` output to determine if NNM processes have completed startup.

In a NNM clustered environment the other command-line options to `ovstart` are not supported.

Note that for fine-grain control of NNM cluster attributes use the `nnmcluster` command directly. The `ovstart` command in a NNM cluster environment is provided for convenience starting NNM using a familiar command.

Parameters

`ovstart` recognizes the following options. Any unrecognized options are reported by a usage message.

`-c`

Produce one line of information about the success or failure of each managed process.

`-d`

Report the important stages in processing, including starting, contacting, and sending the start request to `ovspmd`, and closing the communication channel.

`-o` *`ovspmd_path`*

Specifies that the executable for `ovspmd` is in *`ovspmd_path`* instead of in the default location, *`install_dir`*`\bin`. If `ovspmd` is already running, this option is ignored.

`-v`

Produce several lines of information about the success or failure of each managed process.

`-` *`ovspmd_options`*

Any option not known by `ovstart` is passed to `ovspmd`. Since the `-d` option is valid for both programs, it will be interpreted as an `ovstart` option, and will *not* be passed on to `ovspmd`. Likewise, the `-V` option *will be* passed to `ovspmd` since it is not valid for `ovstart`. If an option is not recognized by either, a usage message will be printed from `ovspmd`, not `ovstart`.

`--`

Terminates the options section of the `ovstart` command line. Any arguments following the comment token (`--`) are interpreted as names of managed processes to start, and passed to `ovspmd`.

RETURN VALUE

In a non NNM cluster environment `ovstart` exits with the status representing the number of object managers from the start list that were *not* started successfully. If all requested managed processes were started successfully, `ovstart` exits with the status `0` (zero).

In a NNM cluster environment `ovstart` always exit immediately with the status `0` (zero).

DIAGNOSTICS

`ovstart` reports certain command-line errors (in particular, too many arguments) and system errors. The messages are prefixed with `ovstart:`, and are intended to be self-explanatory. `ovstart` also outputs error messages received from `ovspmd`. These messages are prefixed with `ovspmd:`. `ovstart` does not treat unrecognized options as errors, but `ovspmd` does.

Note that `ovspmd` can process multiple requests (`ovstart`, `ovstop`, or `ovstatus`) at a time. If any of these commands is being handled, the new request will be queued by type until the previous command has completed.

EXAMPLES

`ovstart`

Request `ovspmd` to start all managed processes configured to start by default. If `ovspmd` is not already

running, start it with no options. Only failures are reported.

```
ovstart -v -V -- ovjboss
```

Request ovspmd to start the ovjboss process, which results in starting the Jboss application server and all of the NNM services that are deployed together within Jboss, after first starting any other managed processes that the ovjboss process depends on. If ovspmd is not already running, start it in verbose mode (-V option). Report program startup, whether successful or not (-v option). Note that the comment token (--) option is necessary so that ovstart does not interpret ovjboss as an argument to the unrecognized -V option.

AUTHOR

ovstart was developed by Hewlett-Packard Company.

FILES

See the nnm.envvars reference page (or the UNIX manpage) for information on using environment variables for the following files:

install_dir\bin\ovstart

install_dir\bin\ovspmd

EXTERNAL INFLUENCES

Environmental Variables

$LANG provides a default value if the internationalization variables, LC_ALL, LC_CTYPE, and LC_MESSAGES are unset, null, or invalid.

If $LANG is unset, null, or invalid, the default value of C (or English_UnitedStates.1252 on Windows) is used.

LC_ALL (or $LANG) determines the locale of all other processes started by ovspmd.

LC_CTYPE determines the interpretation of text as single-byte and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the language in which messages are displayed.

If ovstart is executed, and no ovspmd process is currently running, ovspmd inherits the environment of the executing shell. All managed processes started by ovspmd inherit this same environment.

To change the environment for ovspmd or any managed process, you must restart ovspmd with the correct environment. This requires that all managed processes be stopped (ovspmd does not terminate until all managed processes have been shut down).

As a result, to change the environment for any managed process started from ovstart/ovspmd, you must do the following:

1. Execute `ovstop` with no arguments to shut down all managed processes and `ovspmd`.
2. Set up the correct environment variables.
3. Execute `ovstart` to restart `ovspmd` and any or all managed processes.

NNM Cluster

If a `com.hp.ov.nms.cluster.name` is defined in the `$NnmDataDir/shared/nnm/conf/props/nms-cluster.properties` file, then ovstart will defer startup to the `nnmcluster` command.

SEE ALSO

ovstatus(1), ovstop(1M), ovaddobj(1M), ovdelobj(1M), ovspmd(1M), nnmcluster(1M).

Return to Reference Pages Index

Name

ovstop — stop NNM managed processes

SYNOPSIS

`ovstop` [[-c] [-d] [-v] [*managed_process_names*...]] [[-nofailover|-failover|-cluster]]

DESCRIPTION

`ovstop` stops the NNM managed processes. `ovstop` sends a stop request (`OVS_REQ_STOP`) to the process management process (UNIX operating systems) or service (Windows operating systems), `ovspmd`. If called with one or more *managed_process_name* arguments, it stops the designated managed processes after first stopping any dependent processes. If called with no arguments, or if one of the named arguments is `ovspmd`, it stops all managed processes currently running, including `ovspmd` itself.

When a managed process does not respond to the `ovstop` request within the LRF-specified timeout interval, `ovspmd` forces the process to terminate by sending it termination signals, first `SIGTERM`, then `SIGKILL` (see kill(1)). Note that `ovstop` reports forced termination only if the `-v` or `-c`options are used (for example, `ovstop -v` [*managed_process_name*]). Whenever a managed process times out during a stop request, it is advisable to increase its timeout value. To increase the number of seconds that `ovspmd` waits for a process to respond to an `ovstop` request, follow the instructions in `$NNM_LRF/ov*` (UNIX operating system) or *install_dir*`\lrf\ov*` (Windows operating systems).

Unlike `ovstart`, `ovstop` will *not* start `ovspmd` if it is not already running.

The managed processes are configured by `ovaddobj` from information in Local Registration Files (see lrf(4)). A managed process is named by the first field in the LRF describing it. Like `ovstart`, `ovstop` uses dependency information from the LRF. If other managed processes depend on a managed process that is stopped, `ovspmd` notes their dependency and terminates all appropriate managed processes in reverse LRF dependency order.

`ovstop` must be run by the Windows administrator or UNIX superuser.

If an `OVs_DAEMON` process is configured with a `Stop Command` in its LRF entry, `ovstop` runs the command (see lrf(4)). This feature is used to stop processes that are no longer in contact with `ovspmd`. The `Stop Command` is provided and configured by the developer of the process, if appropriate.

The names of the NNM managed processes that were started by previous ovstart operation can be obtained by running the `ovstatus -c` command.

The `ovstop ovjboss` command would stop the Jboss application server and all of the NNM services deployed together within Jboss. The names of Jboss deployed NNM services can be obtained by running the `ovstatus -v ovjboss` command. The NNM services could only be stopped altogether by running the `ovstop ovjboss` command. It is not supported to stop any of these NNM services individually, independent of the other NNM services.

If `ovstop` is used on a node configured for NNM clustering (see nnmcluster(1M)) then the behavior of ovstop is different than described above. Specifically, ovstop (with no parameters) behaves exactly like the

"nnmcluster -shutdown" command.

In a NNM cluster environment ovstop returns immediately (after sending the NNM cluster a shutdown signal in the background). The nnmcluster command then shuts down NNM processes which might trigger a failover of NNM services to the standby cluster node. Please monitor ovstatus output to determine if NNM processes have completed shutdown.

In a NNM clustered environment the only command-line options recognized by ovstop are -nofailover, -failover, and -cluster.

Note that for fine-grain control of NNM cluster attributes use the nnmcluster command directly. The ovstop command in a NNM cluster environment is provided for convenience shutting down NNM services using a familiar command.

Parameters

ovstop recognizes the options described below. The first argument that is not an option, and any succeeding arguments, are interpreted as names of managed processes to stop, and are passed to ovspmd in the stop request.

-c

Produce one line of information about the success or failure for each managed process.

-d

Report the important stages in its processing, including contacting and sending the stop request to ovspmd, and the closing the communication channel.

-v

Produce several lines of information about the success or failure of each managed process.

-failover

(NNM cluster only) Causes the local NNM node to shutdown NNM processes (if it is the active node) and the NNM cluster process will terminate. At the same time, automatic failover is enabled so that NNM services will transfer to the standby node.

-nofailover

(NNM cluster only) Causes the local NNM node to shutdown NNM processes (if it is the active node) and the NNM cluster process will terminate. At the same time, automatic failover is disabled so that NNM services will not transfer to the standby node.

-cluster

(NNM cluster only) Causes all nodes in the NNM cluster to shutdown. The NNM cluster process on the standby node(s) will be shutdown first, then the active node will stop NNM services, and finally the NNM cluster process on the active node will shutdown.

RETURN VALUE

ovstop exits with a status representing the number of managed processes that were *not* stopped

successfully. If all requested managed processes were successfully stopped, ovstop exits with the status 0 (zero).

DIAGNOSTICS

ovstop reports certain command-line errors (in particular, too many arguments) and system errors. The messages are prefixed with ovstop:, and are intended to be self-explanatory. ovstop also outputs error messages received from ovspmd. These messages are prefixed with ovspmd:. ovstop ignores unrecognized options.

If a managed process is in a PAUSED, PAUSE_ERROR, PAUSE_TIMEOUT, RESUME_ERROR, RESUME_TIMEOUT, or DEPENDENCY_ERR state, it is stopped. However, a warning message is printed to inform you that ovstop was used on a process that was not in a running state.

Note that ovspmd can process multiple requests (ovstart, ovstop, or ovstatus) at a time. If any of these commands is being handled, the new request will be queued by type until the previous command has completed.

AUTHOR

ovstop was developed by Hewlett-Packard Company.

FILES

The environment variables below represent universal pathnames that are established according to your shell and platform requirements. See the nnm.envvars(1) manpage for information on universal pathnames for your platform and shell.

See the nnm.envvars reference page ((or the UNIX manpage) for information about using environment variables for the following files:

Windows: *install_dir*\bin\ovstop

Windows: *install_dir*\bin\ovspmd

UNIX: $NNM_BIN/ovstop

UNIX: $NNM_BIN/ovspmd

EXTERNAL INFLUENCES

Environmental Variables

If a com.hp.ov.nms.cluster.name is defined in the $NnmDataDir/shared/nnm/conf/props/nms-cluster.properties file, then ovstop will defer startup to the nnmcluster command.

$LANG provides a default value if the internationalization variables, LC_ALL, LC_CTYPE, and LC_MESSAGES are unset, null, or invalid.

If $LANG is unset, null, or invalid, the default value of C (or English_UnitedStates.1252 on Windows) is used.

LC_ALL (or $LANG) determines the locale of all other processes started by ovspmd.

LC_CTYPE determines the interpretation of text as single-byte characters, multiple-byte characters, or both; the classification of characters as printable; and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the language in which messages are displayed.

NNM Cluster

If a NNMCLUSTER_NAME is defined in the ov.conf file, then ovstop will defer startup to the nnmcluster command.

SEE ALSO

ovstatus(1), ovstart(1M), ovaddobj(1M), ovdelobj(1M), ovspmd(1M), nnmcluster(1).

Return to Reference Pages Index

Name

pmd — NNM Postmaster service

pmdmgr — NNM Postmaster manager

SYNOPSIS

`pmd` [[-L*size*] [-T*size*] [-S*stack*\;*option*[\;*option*] ...]]

`pmdmgr` [[-L*size*] [-T*size*] [-S*stack*\;*option*[\;*option*]...]]]

DESCRIPTION

`pmd` is the NNM Postmaster service that receives NNM events forwarded from remote NNM 6.X and 7.X management stations and forwards them to the Incident pipeline.

The NNM `OV_EVENT` stack, embedded within pmd, actually performs these services. In addition, `OV_EVENT` logs events to the event logs in *`data_dir`*`\shared\nnm\databases\eventdb`. Logged events can be viewed with `nnmdumpevents`.

`pmdmgr` is the Postmaster service manager. You can use `pmdmgr` to alter stack configurations of the running Postmaster. For example, after you start `pmd`, you can change a stack's trace and log mask with this command. You can alter only general stack options with this command. You can set options specific to a stack (for example, `OV_EVENT`) only at startup.

Only the superuser can use the `pmdmgr` command.

Parameters

`pmd` and `pmdmgr` recognize the following options. Options that are specific to only `pmd` or `pmdmgr` are listed as such. You can use the other options for either command.

`-S`*`stack`*`;`*`option`*`[;`*`option`*`...]`

> Allows you to specify options that are particular to a stack. You can specify general and specific stack options for all stacks by using the `-S` option. For a list of all standardized stack options, see the section called *General Substack Options* of this reference page (and the UNIX manpage). A stack can also have options that are unique to itself.

`-Qt`

> Allows you to truncate the trace files while `pmd` is running. This option is valid only for `pmdmgr`.

`-Ql`

> Allows you to truncate the log files while `pmd` is running. This option is valid only for `pmdmgr`.

`-L`*`size`*

Sets the maximum size of the pmd log files, pmd.log0 and pmd.log1. By default the size of each log file is 500K.

-T*size*

Sets the maximum size of the pmd trace files, pmd.trc0 and pmd.trc1. By default the size of each trace file is 1000K.

Supported Stacks

pmd can be composed of many stacks, where each stack provides a service for pmd. For NNM, the following stacks are included:

OV_EVENT

Receives NNM events forwarded from remote NNM 6.X and 7.X management stations and forwards them to the Incident pipeline.

TRCLOG

Provides the pmd tracing and logging functionality.

General Substack Options

pmd(pmdmgr) recognizes the following options for all stacks. See the -S option (above) that allows you to specify stack options for each stack.

E

Enables a stack. It switches a stack ON. You cannot switch on stacks by using pmdmgr after starting pmd.

D

Disables a stack. It switches a stack OFF. You cannot switch stacks off by using pmdmgr after starting pmd.

T mask

Controls the trace mask for a particular stack. Each stack can have different kinds of tracing and logging enabled. To find out how to OR different types of bits into this mask, see the *Tracing and Logging* section of this reference page (and the UNIX manpage). By default, WARNING, ERROR, and DISASTER messages are logged. You can alter the trace and log mask value by using pmdmgr while pmd is running. The new trace and log mask takes effect immediately.

Tracing and Logging

pmd creates trace and log files in *install_dir*\log\. These are pmd.trc[0-1] and pmd.log[0-1]. The trace files contain all trace and log information. The log files contain only the log information.

pmd traces and logs in a circular fashion, by using two files for wrapping. When the 0 file becomes full, the 0 file is moved to the 1 file, and the old 1 file is truncated and made the 0 file. As a result, you always have

the most current set of log or trace messages. The `0` file is always the current file.

By default, all stacks, as well as the `pmd` itself, log `WARNING`, `ERROR`, and `DISASTER` messages. By default, tracing is not enabled. The `pmd.trcX` files are not created and used until you enable tracing. The trace files are created on demand by turning on some tracing with the `-T` general stack option.

You can turn on additional tracing and logging by altering `pmd.lrf`. To alter a stack's tracing or logging, you need to change its trace mask. You can specify a trace mask for any stack in `pmd`. To create a mask that is a combination of the bits that are listed below, you add the bits together. In effect, you "OR" the bits. The result is then used as the argument to the substack `-T` option.

DISASTERS (0x1)

Log disasters. Disasters should not occur. If they do, contact HP and supply a tracing and logging file to improve the quality of the product.

ERRORS (0x2)

Log errors. These are errors local to `pmd`, and do not include errors that are defined by protocols. CMIP errors that result from a CMISE operation are not logged by setting this bit. Errors such as no route or aborts are logged with this bit.

WARNINGS (0x4)

Log warnings. These are unusual conditions the system administrator may need to know about. These are not necessarily error conditions.

INFORM (0x8)

Log informative messages. These can be messages of various kinds (for example, when signals are received by `pmd`, what the `argv/argc` vectors are that the `pmd` was started with, and so on).

STATE (0x10)

Log state changes. This logging applies mostly to associations and bindings to `pmd`.

HDRIN (0x20)

Trace header information for operations that flow from a stack to `pmd`.

HDROUT (0x40)

Trace header information for operations that flow from `pmd` to a stack.

PDUIN (0x80)

Trace PDU for operations that flow from a stack to `pmd`.

PDUOUT (0x100)

Trace PDU for operations that flow from `pmd` to a stack.

RQT (0x200)

Trace `pmxxx_req` calls that are related to an RQT. These calls operate on RQTs.

MEM (0x400)

Trace memory allocation and de-allocation calls.

CCES (0x800)

Trace calls that are related to CCEs (for example, allocation, freeing, and finding).

TIMERS (0x1000)

Trace timer-related `pm_xxxx_timer` function calls.

STACKCALLS (0x2000)

Trace stack-supplied `SI_XXXX` functions as `pmd` calls them.

OPERATION (0x4000)

Trace `pm_stack_ind`, `pm_stack_cnf`, and `pm_failed_operation`.

ALL_KINDS (0xffffffff)

Log and trace everything.

OV_EVENT Stack Tracing

`OV_EVENT` supports the following trace values. (None are enabled by default.) To create a mask that is a combination of the bits that are listed below, you add the bits together which effectively OR's the bits. The result is then used as the argument to the substack T option, and traces are written to the pmd trace file.

EVENTCONNECTIONS (0x00400000)

Trace all application connections and disconnections to OV_EVENT.

EVENTRECEIPT (0x00800000)

Trace the receipt of each event by OV_EVENT.

EVENTFLOW (0x01000000)

Trace significant milestones as an event flows through and out OV_EVENT. (Includes EVENTQUEUES trace as defined below.)

EVENTFILTERS (0x02000000)

Trace events as they are filtered out; that is, when they are not forwarded to a destination application.

EVENTQUEUES (0x08000000)

Trace events as they are queued for a busy application and when they are removed from the queue the application is listening.

OV_EVENT Specific Stack Options

The OV_EVENT stack supports the following options:

b*size*

Sets the total size of the OV_EVENT log files to the given size, in megabytes (default size is 16 MB). Minimum size is 1 megabyte. Once the size is exceeded, oldest events will be dropped from the event logs.

n

Disables OV_EVENT logging to the event log files. The default is to log.

q*num*

Specifies the maximum number of events which can be queued for a connected application. When this number is exceeded, OV_EVENT will disconnect from the application. Events are queued when they cannot be sent to an application (receipt is blocked because the application is busy elsewhere). The default maximum is 4096 events. Valid values range from 0 to 65536.

DIAGNOSTIC

For help with problems related to starting pmd, refer to the troubleshooting section of *HP OpenView Managing Your Network with NNM*.

EXAMPLES

Switch on all possible (general and stack-specific) tracing for the OV_EVENT stack. You want to see all inbound and outbound PDUs to and from this stack, as well as all other information. Change the pmd.lrf file to the following:

```
pmd:pmd:
OVs_YES_START::-SOV_EVENT;T0xffffffff:OVs_WELL_BEHAVED::
```

For these changes to take effect, first stop the NNM platform using the ovstop command. Second, use the ovdelobj command to delete the old pmd.lrf entry. Third, use the ovaddobj command to add the modified pmd.lrf entry. Finally, use the ovstart command to restart the platform.

To change the OV_EVENT trace and log mask after pmd process is running, use the following command:

```
pmdmgr -SOV_EVENT\;T0xf
```

This command causes INFORM messages to be logged, as well as DISASTER, WARNING, and ERROR messages.

AUTHOR

pmd was developed by Hewlett-Packard Company.

FILES

See the `nnm.envvars` reference page for information on using environment variables for the following files:

data_dir`\shared\nnm\lrf\pmd.lrf`

data_dir`\shared\nnm\databases\eventdb`

install_dir`\log\pmd.log[01]`

install_dir`\log\pmd.trc[01]`

SEE ALSO

nnmdumpevents(1), ovstatus(1), ovstart(1M), ovstop(1M), ovspmd(1M).

Return to Reference Pages Index