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

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

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

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`*

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
```

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`

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

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

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