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 Notices

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)

This product includes software developed by The Legion Of The Bouncy Castle. (http://www.bouncycastle.org)

This product contains software developed by Trantor Standard Systems Inc.. (http://www.trantor.ca)

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 nnmincidentcfg.ovpl nnmldap.ovpl nnmlicense.ovpl nnmloadattributes.ovpl nnmloadmib.ovpl nnmloadnodegroups.ovpl nnmloadseeds.ovpl nnmmanagementmode.ovpl nnmnetdeletenodeattrs.ovpl nnmnetloadnodeattrs.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 nnmsetofficialfqdn.ovpl nnmsnmpbulk.ovpl nnmsnmpget.ovpl	disco.NoVLANIndexing disco.SkipXdpProcessing hostnolookup.conf ipnolookup.conf logging.properties macdedupexceptions.txt nnm.ports nnm.properties ldap.properties nnmtrapd.conf trapFilter.conf UnnumberedNodeGroup.conf UnnumberedSubnets.conf

Name

nnmcluster — start NNMi cluster services

SYNOPSIS

`nnmcluster` [-disable|-enable] [-display] [-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

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.

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

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

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.

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

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.

`-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

Name

nnmsnmpwalk.ovpl — query a node using multiple SNMP 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*]

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.

`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

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

logging.properties — file containing configurable logging properties used by the `NMSLogManager` service to control how NNMi uses the `java.util.logging` framework to produce log files.

SYNOPSIS

`logging.properties`

DESCRIPTION

`logging.properties` is a file used by the `NMSLogManager ovjboss` service to configure how NNMi produces log files using the `java.util.logging` framework. After you make changes to this file, you must restart the `ovjboss` servuce, or run the `$NnmInstallDir/support/nnmrereadlogging.ovpl` script.

NOTE: This document does not explain the `java.util.logging` framework. To learn more about this framework, go to http://download.oracle.com/javase/1.4.2/docs/api/java/util/logging/package-summary.html

The format of the `logging.properties` file conforms to the standard format required by the `java.util.logging` framework, and adds some custom extensions used by NNMi to support NNMi logging requirements. In addition to the basic `java.util.logging` framework logging properties, NNMi adds the following `name=value` entries to the `logging.properties` file:

`<package.name>.<LogFileHandler>.count`

Specifies the number of archive log files NNMi should keep when `ovjboss` restarts. The default value is 2.

`<package.name>.<LogFileHandler>.limit`

Specifies the size (in bytes) the active log file should become before being rolled to a new active log file. The default value is 1048576 (1MB).

`<package.name>.<LogFileHandler>.pattern`

Specifies the pattern to use in naming log files for the `LogFileHandler`. The default value is `<logBaseName>.%g.%u.log`.

Where:

- `%g` corresponds to the current count of archived log sets (see count above).
- `%u` is controlled by `java.util.logging` based on file locking. The "u" in `%u` stands for *unique*. This value is only incremented if the current file name is locked. When that happens, the process increments `%u` until it can get a file lock, so that a new log file can be created.

  NOTE: While `%g` reflects a limit on the number of archived sets of logs, an increment of `%u` shows that the file is currently locked. In the `logging.properties` file, if you set 5 for count and 1MB for limit, you can get up to five sets of archive log files. Each set has as many 1MB log files as are required to store the logging data.

As with standard `java.util.logging` framework properties files, you can specify default log levels, parent logger relationships, log formatters, and other log handlers to use. If you are unsure about changing any logging setting, **it's a good idea to check with support to make sure you make appropriate changes**.

EXAMPLES

If you want to keep five sets of NNMi log files and roll the active logs by size approximately every 5MB, edit the `logging.properties` file as follows:

```
com.hp.ov.nms.admin.log.NnmMainFileHandler.limit = 5000000
com.hp.ov.nms.admin.log.NnmMainFileHandler.count = 5
```

Assuming you have a very busy system and you restart it once a week, you could get a set of files that looks something like this:

```
nnm.0.0.log Note: the current log file
nnm.1.0.log Note: this may have been archived due to a process restart
completed this morning
nnm.2.0.log Note: this may have been caused by hitting a file size limit of 5
MB
nnm.3.0.log Note: this may have been archived due to a process restart
completed this morning
nnm.4.0.log Note: this may be an archive of the very first log file, and may
have been created when the
first log file surpassed a file size limit of 5 MB
additional log entries...
```

AUTHOR

`logging.properties` was developed by Hewlett-Packard Company.

FILES

Windows: `%NNM_DATA%\shared\nnm\conf\ovjboss\logging.properties`

UNIX: `$NNM_DATA/shared/nnm/conf/ovjboss/logging.properties`

NNMi specific `java.util.logging` framework properties used by the `NMSLogManager` service.

SEE ALSO

ovjboss(1M).

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

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

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`*> [-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

`-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`.

`-node <nodename|IP Address>`

Target node name or IP address.

`-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
```

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(1M).

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(1M).

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