Netra High Availability Suite Foundation Services 2.1 6/03 Reference Manual

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–1773–11
September 2004

040825@9495

Contents

Preface

The *Netra High Availability Suite Foundation Services 2.1 6/03 Reference Manual* describes the tools, API, configuration files, and drivers delivered in the Foundation Services. Both novice users and those familar with Foundation Services can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question "What does it do?" The man pages in general comprise the reference manual. They are not intended to be a tutorial.

How This Book Is Organized

The following contains a brief description of each man page section and the information it references:

- Section 1M describes, in alphabetical order, the tools and commands that are used chiefly for installation and administration purposes.
- Section 3CMM describes, in alphabetical order, the functions in the Cluster Membership Manager (CMM) library. The CMM API provides applications with information about what nodes are in the cluster and the roles of nodes in the cluster.
- Section 4 describes, in alphbetical order, the installation and Foundation Services configuration files.
- Section 7D describes the Carrier Grade Transport Protocol (CGTP) driver.

Below is a generic format for man pages. The man pages of each manual section generally follow this format. All headings may not appear in each man page. See the `intro` pages for more information and detail about each section, and `man`(1) for more information about man pages in general.

NAME	This section gives the names of the command, function, or configuration file documented, followed by a brief description of what they do.

SYNOPSIS

This section shows the syntax of the command, function, or configuration file. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

[ ]	Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.
. . .	Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "`filename` . . .".
|	Separator. Only one of the arguments separated by this character can be specified at a time.
{ }	Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

DESCRIPTION

This section defines the functionality and behavior of the command, function, or configuration file. Thus it describes concisely what the command does. It does not discuss options or cite examples. Interactive commands, subcommands, requests, macros, and functions are described under USAGE.

OPTIONS

This secton lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.

OPERANDS

This section lists the command operands and describes how they affect the actions of the command.

OUTPUT

This section describes the output – standard output, standard error, or output files – generated by the command.

RETURN VALUES	If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.
ERRORS	On failure, most functions place an error code in the global variable `errno` indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.
USAGE	This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality: Commands Modifiers Variables Expressions Input Grammar
EXAMPLES	This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as `example%`, or if the user must be superuser, `example#`. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections.
ENVIRONMENT VARIABLES	This section lists any environment variables that the command or function affects, followed by a brief description of the effect.
EXIT STATUS	This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions.

FILES	This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.
ATTRIBUTES	This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes`(5) for more information.
SEE ALSO	This section lists references to other man pages, in-house documentation, and outside publications.
DIAGNOSTICS	This section lists diagnostic messages with a brief explanation of the condition causing the error.
WARNINGS	This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.
NOTES	This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.
BUGS	This section describes known bugs and, wherever possible, suggests workarounds.

Related Books

You will require some of the following books from the Foundation Services documentation set:

- *Netra High Availability Suite Foundation Services 2.1 6/03 Overview*
- *Netra High Availability Suite Foundation Services 2.1 6/03 Glossary*
- *What's New in Netra High Availability Suite Foundation Services 2.1 6/03*
- *Netra High Availability Suite Foundation Services 2.1 6/03 Quick Start Guide*
- *Netra High Availability Suite Foundation Services 2.1 6/03 Hardware Guide*
- *Netra High Availability Suite Foundation Services 2.1 6/03 Custom Installation Guide*
- *Netra High Availability Suite Foundation Services 2.1 6/03 Cluster Administration Guide*
- *Netra High Availability Suite Foundation Services 2.1 6/03 Troubleshooting Guide*
- *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*
- *Netra High Availability Suite Foundation Services 2.1 6/03 NMA Programming Guide*

- *Netra High Availability Suite Foundation Services 2.1 6/03 Reference Manual*
- *Netra High Availability Suite Foundation Services 2.1 6/03 Standalone CGTP Guide*
- *Netra High Availability Suite Foundation Services 2.1 6/03 Release Notes*
- *Netra High Availability Suite Foundation Services 2.1 6/03 README*

Introduction

NAME

Intro – introduction to daemons, system maintenance commands, and installation tool commands

DESCRIPTION

This section lists the daemons, system maintenance commands, and installation tool commands.

Daemons

This section lists the daemons.

`nhcmmd`(1M)	Cluster Membership Manager (CMM) daemon
`nhcrfsd`(1M)	Reliable NFS supervisory daemon
`nhpmd`(1M)	Daemon Monitor daemon
`nhprobed`(1M)	Probe link and node accessibility daemon
`nhnsmd`(1M)	Node State Manager daemon
`nma`(1M)	Node Management Agent daemon
`nhwdtd`(1M)	Watchdog Timer daemon

Maintenance Commands

This section lists the system maintenance commands.

`nhadm`(1M)	Foundation Services administration tool
`nhcmmqualif`(1M)	Command to qualify the current node as master
`nhcmmrole`(1M)	Command to obtain the role of the node
`nhcmmstat`(1M)	Command to display information about peer nodes, trigger a switchover, or force the qualification of a master-eligible node
`nhcrfsadm`(1M)	Command for Reliable NFS administration
`nhenablesync`(1M)	Command to trigger synchronization
`nhpmdadm`(1M)	Daemon Monitor administration tool
`nhpmdadmwrapper`(1M)	Command to configure Daemon Monitor values
`nhsched`(1M)	Command to display the scheduling parameters of the Foundation Services processes

nhinstall Command

This section contains the `nhinstall` command.

`nhinstall`(1M)	Foundation Services installation and configuration tool

SMCT Commands

Caution – Do not use the SMCT commands with the current patch level of the Foundation Services product.

This section lists the Software Management and Configuration Toolkit (SMCT) commands.

`flconfig`(1M)	Add user defined configuration data to a flash archive
`flcreate`(1M)	Create a generic flash archive from the node group software
`fldeploy`(1M)	Generate a deployable flash archive and Solaris JumpStart environment
`flinstall`(1M)	Generate a Solaris JumpStart environment for a diskfull node group
`nhsmctsetup`(1M)	Create the SMCT environment
`slconfig`(1M)	Add user defined configuration data to the software load
`slcreate`(1M)	Prepare the data for a generic flash archive
`sldelete`(1M)	Delete a software load
`sldeploy`(1M)	Generate configuration files for a software load
`slexport`(1M)	Copy software load data to an export directory

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

NAME

Intro – introduction to the Cluster Membership Manager API functions

DESCRIPTION

This section describes the Cluster Membership Manager (CMM) Application Programming Interface (API) functions.

LIST OF FUNCTIONS

`cmm_cmc_filter`(3CMM)
Define notification filtering

`cmm_cmc_register`(3CMM)
Register to receive notifications

`cmm_cmc_unregister`(3CMM)
Unregister to stop receiving notifications

`cmm_config_reload`(3CMM)
Reload the cluster node table

`cmm_connect`(3CMM)
Prepare or test a connection to the Cluster Membership Manager

`cmm_disconnect`(3CMM)
Close a connection between current calling process and the `nhcmmd` daemon

`cmm_master_getinfo`(3CMM)
Retrieve master node information

`cmm_vicemaster_getinfo`(3CMM)
Retrieve the vice–master node information

`cmm_mastership_release`(3CMM)
Trigger a switchover

`cmm_member_getall`(3CMM)
Retrieve cluster membership information

`cmm_member_getcount`(3CMM)
Retrieve number of nodes in the cluster

`cmm_member_getinfo`(3CMM)
Retreive information about a peer node

`cmm_member_isdesynchronized`(3CMM)
Interpret the synchronization status of a master-eligible node

`cmm_member_isdisqualified`(3CMM)
Remove peer node

`cmm_member_iseligible`(3CMM)
Determine if a node is master-eligible

`cmm_member_isexcluded`(3CMM)
Determine if a node is excluded from the cluster

`cmm_member_isfrozen`(3CMM)
Determine if a node is frozen

`cmm_member_ismaster`(3CMM)
Determine if a node is the master

`cmm_member_isvicemaster`(3CMM)
Determine if a node is the vice master

`cmm_member_isoutofcluster`(3CMM)
Determine if a node is not participating in the cluster

`cmm_member_isqualified`(3CMM)
Determine if a node is eligible to be master

`cmm_membership_remove`(3CMM)
Remove a peer node from the cluster

`cmm_member_seizequalif`(3CMM)
Force requalification of master-eligible node

`cmm_member_setqualif`(3CMM)
Give a new level of qualification to a node

`cmm_node_getid`(3CMM)
Retrieve ID of a node

`cmm_notify_dispatch`(3CMM)
Dispatch cluster membership change messages

`cmm_notify_getfd`(3CMM)
Receive cluster membership change messages by getting file descriptor associated with registration

`cmm_potential_getinfo`(3CMM)
Retrieve information about a peer node even if it has the `OUT_OF_CLUSTER` role.

`cmm_strerror`(3CMM)
Get error message string

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

NAME

Intro – introduction to the Foundation Services configuration files

DESCRIPTION

The following types of configuration files are included in this section:

- Configuration files for the `nhinstall` tool
- Configuration files for the cluster
- Configuration files for the Node Management Agent (NMA)

LIST OF FILES

`nhinstall` Configuration Files

The following files enable you to configure the `nhinstall` tool to install the Foundation Services on the cluster:

`addon.conf`(4)
`nhinstall` configuration file to install additional patches and Solaris packages

`cluster_definition.conf`(4)
`nhinstall` configuration file to define the cluster

`env_installation.conf`(4)
`nhinstall` configuration file defining the installation environment

`nodeprof.conf`(4)
file that permits the customization of Solaris installation

`diskless_nodeprof.conf`(4)
file that permits the customization of Solaris installation on diskless nodes

Cluster Configuration Files

The following are the cluster configuration files. These files are installed and configured by the `nhinstall` tool. Only if you are installing the Foundation Services manually, you must modify these files:

`cluster_nodes_table`(4)	Node management file
`nhadmsync.conf`(4)	List of nonreplicated files and the differences between them
`nhfs.conf`(4)	Foundation Services configuration file
`target.conf`(4)	Basic node configuration file
`nhpmd.conf`(4)	Specification of number of retries for a daemon monitored by PMD if this number is not the default

NMA Configuration Files

The following files enable you to program the NMA to monitor your cluster:

`nma.notifs.txt`(4)	NMA configuration file for SNMP trap notifications
`nma.params.txt`(4)	NMA configuration file for SNMP parameters
`nma.properties`(4)	NMA configuration file defining NMA properties
`nma.security`(4)	NMA configuration file for SNMP security
`nma.targets.txt`(4)	NMA configuration file for SNMP trap targets

SMCT Configuration Files

Caution – Do not use the SMCT configuration files with the current patch level of the Foundation Services product.

The following files enable you to define your cluster configuration for the SMCT:

`cluster.conf`(4)	SMCT file that contains a logical view of the cluster in terms of nodes, node groups, domains and services
`install-server.conf`(4)	SMCT file to configure installation server network
`machine.conf`(4)	SMCT file that describes the cluster in terms of hardware elements, disk layout, and file system definitions.
`master-system.conf`(4)	SMCT file to configure the prototype machine network
`network.conf`(4)	SMCT file that describes the network parameters for the target cluster
`software.conf`(4)	SMCT file that describes the software packages and patches to be deployed to each node group
`userapp.conf`(4)	SMCT file containing a structured list of user defined configuration data to be installed at cluster startup

ATTRIBUTES

See `attributes`(5) for a discussion of the attributes listed in this section.

Maintenance Commands

NAME

flconfig – SMCT command to add user-defined configuration data to a flash archive

SYNOPSIS

`flconfig -n` *swl-name* **`-v`** *swl-version* **`-g`** *node-group-name* **`-f`** *flash-archive* **`[-e`** *export-dir*`]` **`[-c`** *config-dir*`]` **`[-l`** *logfile*`]` **`[-V`** *verbosity-level*`]` **`[-h]`**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `flconfig` command adds user-defined configuration data to a specified flash archive. This command uses data from the export directory to create a configuration section that contains all the user application data and application data installation scripts for the node group specified as a parameter. The configuration section is added to the flash archive for the node group.

Configuration sections are also created for the diskless node groups embedded in the master-eligible node groups.

You must be superuser to run the `flconfig` command.

OPTIONS

The options that you can use with the `flconfig` command are:

`-n` *swl-name*	Name of the software load to process. This is an ASCII string.
`-v` *swl-version*	Version of the software load to process. This is an ASCII string.
`-g` *node-group-name*	Name of the node group associated to the flash archive. This is an ASCII string defined in the file `cluster.conf`.
`-f` *flash-archive*	Absolute path and file name of the flash archive. If only the flash archive name is specified, the flash archive is expected to be located in the default directory `SMCT_FLASH_DIR`, which is defined in the `smct.env` file.
`-e` *export-dir*	Directory where the software load data has been copied to by the `slexport` command. The default directory is specified by the SMCT variable `SMCT_EXPORT_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-c` *config-dir*	Directory where the configuration files are located. This option overrides the configuration directory specified by the SMCT variable `SMCT_DEFAULT_CONFIG_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.

`-l` *log-file*	Name of the file that stores information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`flcreate`(1M), `fldeploy`(1M), `flinstall`(1M), `nhsmctsetup`(1M), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M), `slexport`(1M)

NAME

flcreate – SMCT command to create a generic flash archive from the node group software

SYNOPSIS

`flcreate -f` *flash-archive* **`-n`** *swl-name* **`-v`** *swl-version* **`-g`** *node-group-name* **`[-e`** *export-dir*`]` **`[-c`** *config-dir*`]` **`[-l`** *logfile*`]` **`[-V`** *verbosity-level*`]` **`[-h]`**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `flcreate` command creates a generic flash archive from the node group software that is installed on the prototype machine. This command is run once for each master-eligible and dataless node group, as each node group is processed individually.

The `flcreate` command must be run as a superuser.

The following configuration files are used by `flcreate` to configure the network between the prototype machine and the installation server. They must be completed before running the `flcreate` command:

- `install-server.conf`

  For more information on the `install-server.conf` file, see the `install-server.conf`(4) man page.
- `master-system.conf`

  For more information on the `master-system.conf` file, see the `master-system.conf`(4) man page.

OPTIONS

The options that you can use with the `flcreate` command are:

`-f` *flash-archive*	Absolute path and file name of the flash archive. If only the name of the flash archive is provided, the flash archive is created in the default directory specified by the SMCT variable `SMCT_FLASH_DIR`. This is an ASCII string.
`-n` *swl-name*	Name of the software load to process. This is an ASCII string.
`-v` *swl-version*	Version of the software load to process. This is an ASCII string.
`-g` *node-group-name*	Name of the nodes group associated to the flash archive. This is an ASCII string defined in the file `cluster.conf`.
`-e` *export-dir*	Directory where the software load data has been copied to by the `slexport` command. This option overrides the export directory specified by the SMCT variable `SMCT_EXPORT_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.

`-c` *config-dir*	Directory where the cluster configuration files are located. This option overrides the configuration directory specified by the SMCT variable `SMCT_DEFAULT_CONFIG_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-l` *log-file*	Name of the file that is sent information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`flconfig`(1M), `fldeploy`(1M), `flinstall`(1M), `install-server.conf`(4), `master-system.conf`(4), `nhsmctsetup`(1M), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M), `slexport`(1M)

NAME

fldeploy – SMCT command to generate a deployable flash archive and Solaris JumpStart environment

SYNOPSIS

`fldeploy -f` *flash-archive* **`-n`** *swl-name* **`-v`** *swl-version* **`-g`** *node-group-name* **`[-e`** *export-dir*`]` **`[-j`** *jumpstart-dir*`]` **`[-c`** *config-dir*`]` **`[-l`** *logfile*`]` **`[-V`** *verbosity-level*`]` **`[-h]`**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `fldeploy` command does the following:

- Creates a deployable flash archive for the node group specified by *node-group-name*.
- Generates a Solaris JumpStart environment for each node of the node group.
- Adds the exported data from the `sldeploy` command to the generic or configured flash archive. The exported data are Foundation Services configuration files for services such as Reliable NFS and the Cluster Membership Manager (CMM), and includes the configuration file, `nhfs.conf`.

The `fldeploy` command must be run as a superuser.

Configure the `install-server.conf` configuration file before running the `fldeploy` command. For more information on the `install-server.conf` file, see the `install-server.conf`(4) man page.

OPTIONS

The options that you can use with the `fldeploy` command are:

`-f` *flash-archive*	Absolute path and file name of the flash archive. If only the name of the flash archive is provided, the flash archive is created in the default directory specified by the SMCT variable `SMCT_FLASH_DIR`. This is an ASCII string.
`-n` *swl-name*	Name of the software load to process. This is an ASCII string.
`-v` *swl-version*	Version of the software load to process. This is an ASCII string.
`-g` *node-group-name*	Name of the node group for which the Solaris JumpStart environments are created. An environment is created for each node of the node group. The *node-group-name* is an ASCII string defined in the file `cluster.conf`.
`-e` *export-dir*	Directory where the software load data has been copied to by the `slexport` command. This option overrides the export directory specified by the SMCT variable `SMCT_EXPORT_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.

`-j` *jumpstart-dir*	Directory where the Solaris JumpStart environment will be created. This option overrides the Solaris JumpStart directory specified by the SMCT variable `SMCT_JUMPSTART_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-c` *config-dir*	Directory where the cluster configuration files are located. This option overrides the configuration directory specified by the SMCT variable `SMCT_DEFAULT_CONFIG_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-l` *log-file*	Name of the file that is sent information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`flconfig`(1M), `flcreate`(1M), `flinstall`(1M), `install-server.conf`(4)`nhsmctsetup`(1M), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M), `slexport`(1M)

NAME

flinstall – SMCT command to generate Solaris JumpStart environments for master-eligible and dataless node groups

SYNOPSIS

`flinstall -n` *swl-name* **`-v`** *swl-version* **`-g`** *node-group-name* **`[-j`** *jumpstart-dir***`]** **`[-e`** *export-dir***`]** **`[-c`** *config-dir***`]** **`[-l`** *logfile***`]** **`[-V`** *verbosity-level***`]** **`[-h]`**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `flinstall` command generates a Solaris JumpStart environment for each master-eligible and dataless node group to be installed on the prototype machine. The `flinstall` command uses the data generated by the `slexport` command.

The `flinstall` command must be run as a superuser.

The following configuration files are used by `flinstall` to configure the network between the prototype machine and the installation server. They must be completed before running the `flinstall` command:

- `install-server.conf`

  For more information see the `install-server.conf`(4) man page.
- `master-system.conf`

  For more information see the `master-system.conf`(4) man page.

OPTIONS

The options that you can use with the `flinstall` command are:

`-n` *swl-name*	Name of the software load to process. This is an ASCII string.
`-v` *swl-version*	Version of the software load to process. This is an ASCII string.
`-g` *node-group-name*	Name of the node group for which the Solaris JumpStart environment is created. An environment is created for each node of the node group. The *node-group-name* is an ASCII string defined in the file `cluster.conf`.
`-j` *jumpstart-dir*	Directory where the Solaris JumpStart environment is to be created. This option overrides the Solaris JumpStart directory specified by the SMCT variable `SMCT_JUMPSTART_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-e` *export-dir*	Directory to which the software load data will be exported. This option overrides the export directory specified by the SMCT variable `SMCT_EXPORT_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.

`-c` *config-dir*	Directory where the configuration files are located. This option overrides the configuration directories specified by the SMCT variables `SMCT_DEFAULT_CONFIG_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-l` *log-file*	Name of the file that is sent information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`flconfig`(1M), `flcreate`(1M), `fldeploy`(1M), `install-server.conf`(4), `master-system.conf`(4), `nhsmctsetup`(1M), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M), `slexport`(1M)

NAME

nhadm – cluster administration tool

SYNOPSIS

/opt/SUNWcgha/sbin/nhadm [-d *data-file* **] copy [** *file* **...]**

/opt/SUNWcgha/sbin/nhadm [-f *file* **] [-s] [-v] check [** *stage* **]**

/opt/SUNWcgha/sbin/nhadm [-f *file* **] [-s] [-v] display**

/opt/SUNWcgha/sbin/nhadm -h

/opt/SUNWcgha/sbin/nhadm [-s] [-v] confshare [
shared_package_directory **]**

/opt/SUNWcgha/sbin/nhadm [-s] [-v] [-y *file* **] synccheck**

/opt/SUNWcgha/sbin/nhadm [-s] [-v] [-y *file* **] syncgen**

/opt/SUNWcgha/sbin/nhadm [-z] [html | text]

DESCRIPTION

The `nhadm` tool provides a suite of tools to check the installation and configuration of the Foundation Services software and its prerequisite products. You must log in as superuser to use this command.

OPTIONS

The options that you can use with the `nhadm` tool are as follows:

`-d | --data`	Specifies the name of the file that lists the files to be copied
`-f | --fsconf`	Specifies the name of the Foundation Services configuration file. If this option is not used, the default file is `/etc/opt/SUNWcgha/nhfs.conf`. For information on this file, see the `nhfs.conf`(4) man page.
`-h`	Displays a help screen.
`-s | --silent`	Runs in silent mode. When using the `nhadm check` command in this mode, the tests being run are not displayed. Only the errors encountered by these tests are displayed.
`-v | --verbose`	Runs in verbose mode. In this mode, traces are displayed when performing operations. The level of detail provided in the traces increases every time this option is added.
`-y | --syncfile`	Specifies the name of the file that lists the nonreplicated files that you want to compare. The default is `/SUNWcgha/remote/etc/nhadmsync.conf`. For more information on this file, see the `nhadmsync.conf`(4) man page.
`-z | --err [ html | text ]`	Prints messages corresponding to all the error scenarios tested by `nhadm` and provides explanations for these errors in html or text form (html is the default). You can

use this command to help understand error messages generated by `nhadm` when you test a node or cluster.

COMMANDS

The commands and stages available with the `nhadm` tool are the following:

`check`

Verifies that the prerequisite software and hardware are correctly installed and configured, that your cluster has started correctly, and that all peer nodes are accessible. The tests run by `nhadm check` are broken down into the following stages. You can run an individual stage that suits the task you are performing by typing `nhadm check` *stage* or you can run all three stages by typing `nhadm check`.

- `installation`

  Checks that the correct version of the Solaris operating system is installed. This command also checks that the Foundation Services packages and necessary patches are present and installed correctly.

- `configuration`

  Checks that the configuration files required before starting up the Foundation Services are present and of the correct format. Also checks that the configuration has been performed successfully.

- `starting`

  Tests the node accessibility and disk replication on a running cluster.

Note – If a stage is not specified, all the stages are run.

`confshare`

A file is required for patching shared packages installed on the cluster. The `confshare` command creates this file by copying `/var/sadm/system/admin/INST_RELEASE`, the file for local packages, to the shared package repository. By default, `confshare` creates the `/var/sadm/system/admin/INST_RELEASE` file in the `/SUNWcgha/local/export/services` shared package directory.

	If your shared packages are not installed in `/SUNWcgha/local/export/services`, you must alter the location where `/var/sadm/system/admin/INST_RELEASE` is created to match your shared package repository. For example: `/opt/SUNWcgha/sbin/nhadm confshare [` *shared_package_directory* `]`
`copy`	Copies files from the master node to the vice-master node. Files listed can be passed as an argument or listed in the *data-file* file.
`display`	Prints information to the console of the node on which it is run. The displayed information includes local and shared packages installed, node and cluster IDs, network interface information, local and mounted file systems, mount points, and shared partition information. This command can help you diagnose problems by listing the current node configuration.
`synccheck`	Uses the Solaris `diff` command to compare the files listed in `/SUNWcgha/remote/etc/nhadmsync.conf` and prints a warning on the console when the files are not identical on the master-eligible nodes. You must include files in `/SUNWcgha/remote/etc/nhadmsync.conf` that are on the master and the vice-master but are not replicated on a shared file system. The compared files can include Solaris files and Foundation Services files. For more information on the `diff` command, see `diff`(1).
`syncgen`	Accepts the differences between the two nodes found by `synccheck` for each file listed in `/SUNWcgha/remote/etc/nhadmsync.conf`.

EXTENDED DESCRIPTION

The `nhadm` tool enables you to verify the status of your cluster. You should use the `nhadm` tool as part of regular maintenance or after you change a cluster configuration in any way.

The `nhadm display` command prints the current status of a node to the console. Further cluster status information can be obtained by running the `nhcmmstat` command, as described in `nhcmmstat`(1M).

The nhadm tool displays an ok message for every check that passes. If any of the tests performed by nhadm check fail, an error message is displayed on the console. The error message describes the error and identifies the command that has failed, or the likely problem area. For an explanation of the possible error messages, use the nhadm -err option.

In a correctly functioning cluster, replicated files are the same on the master node and the vice-master node. Some system files cannot be placed on shared file systems but must contain the same information on both master-eligible nodes. You can list the differences between nonreplicated files and manage the differences between these files using the nhadm synccheck and nhadm syncgen commands.

The list of nonreplicated files compared by nhadm synccheck is shared between the master node and the vice-master node. A template for this file is installed at /opt/SUNWcgha/config.standard/adm/nhadmsync.conf.template. The default location of this shared file is /SUNWcgha/remote/etc/nhadmsync.conf. For further information on the nhadmsync.conf file, see the nhadmsync.conf(3) man page.

EXAMPLES

This section gives examples of how to use the nhadm tool and its commands.

Using nhadm check

This section contains examples of using nhadm check installation, nhadm check conf and nhadm check starting.

Note that the checking stages might vary, depending on the node configuration. For example, the disk check mechanism changes if you are using the Solaris Volume Manager. Therefore, the following examples are guidelines only.

EXAMPLE 1 To Verify Software Installation

After installing the hardware and software, log in to the machine you want to examine and run the nhadm check installation command:

```
# nhadm check installation
```

The nhadm tool verifies that:

- All required software packages and patches are installed
- The Solaris operating system and patches have the correct version
- The same MAC address is not used twice

EXAMPLE 2 To Verify Software Configuration

When the Foundation Services software has been configured, log in to the peer node you want to examine, and run the nhadm check configuration command. This command checks that the configuration files that are required before starting the Foundation Services have been correctly configured.

```
# nhadm check configuration
```

EXAMPLE 2 To Verify Software Configuration *(Continued)*

This command tests the following:

- The cluster definition files are present and in the correct format.
- The network configuration is correctly defined in `/etc/hostname` and `/etc/hosts`.
- The boot-device is configured to be `disk` for master-eligible nodes and `net` for other peer nodes.

  For a Netra 20 peer node, boot-device is `*disk*` where `disk` in this case cannot match `disk` for the OpenBoot PROM.
- The auto-boot option is set to `true`.
- The local-mac-address for this node is set to `true`.
- The root file system is defined in `/etc/vfstab` and the partitions listed in `/etc/vfstab` exist.
- The `/etc/inet/dhcpsvc.conf` file is present. This test is optional and is only run if the Reliable Boot Service package has been installed.

EXAMPLE 3 To Verify Cluster Network and Disk Replication

Use `nhadm check starting` to list any cluster network, or disk replication problems, by logging in to a peer node in a running cluster and typing:

```
# nhadm check starting
```

The `nhadm` tool verifies that:

- Each node interface exists and is functioning correctly
- Each peer node is accessible from the current node
- The shared file systems are replicated

EXAMPLE 4 To Verify the Cluster Node Table Configuration When the Master Node Disk and Vice-Master Node Disk Are Not Synchronized

Log in to a master-eligible node on which you want to verify the file and create the `/etc/opt/SUNWcgha/not_configured` file on the node. Boot the node and type:

```
# nhadm -c /SUNWcgha/local/export/data/etc/cluster_nodes_table check
```

By default, `nhadm` uses the shared file system to access the `cluster_nodes_table` file. When Reliable NFS is not running, as in this case, the file system containing the `cluster_nodes_table` file is not mounted and exported. By specifying the option `-c` and the local path to the `cluster_nodes_table` file, you are forcing `nhadm` to use the local path.

The target `check` can be replaced by `check installation`, `check configuration`, or `check starting`.

Comparing Nonreplicated Files

This section provides examples for using the `nhadm syncheck` and `nhadm syncgen` commands.

To use these commands, both master-eligible nodes must have remote access to each other. To enable this, make sure that the CGTP address of the other master-eligible node is set in the `.rhosts` file on each master-eligible node. For example:

On node `cgtp10` the `/.rhosts` file must contain the CGTP address of the other master-eligible node, `cgtp11`:

```
cgtp11 root
```

On node `cgtp11`, the `/.rhosts` file should contain the CGTP address of the other master-eligible node, `cgtp10`:

```
cgtp10 root
```

This enables Reliable NFS to perform `rsh` between the master eligible nodes.

To use the `nhadm syncheck` command, you must specify the nonreplicated files that you want to compare. By default, you specify the list of files in `/SUNWcgha/remote/etc/nhadmsync.conf`. The `nhadm syncheck` command compares the copies of these files on the master and the vice-master nodes, printing any differences to the console. You can accept the differences using `nhadm syncgen`. Accepted differences are not printed to the console when you run `nhadm syncheck` again.

You can change the name and location of the file that stores the list of nonreplicated files to be compared. You can also have several files containing different lists of nonreplicated files.

EXAMPLE 5 To Specify the Nonreplicated Files to be Compared

Create the `/SUNWcgha/remote/etc/` directory. Copy the template file `/opt/SUNWcgha/config.standard/adm/nhadmsync.conf.template` to `/SUNWcgha/remote/etc/nhadmsync.conf`.

Add the names of the files to be compared to the `nhadmsync.conf` file. The files that you add should have the following criteria:

- The files exist on both master-eligible nodes
- The files are not replicated on a shared file system

Use the following syntax for all entries in the file:

`FILE=`*filename*

For further information on the `nhadmsync.conf` file see the `nhadmsync.conf`(4) man page.

EXAMPLE 6 To Determine the Differences Between Nonreplicated Files

Type:

```
# nhadm synccheck
```

This command compares the differences between the files listed in `nhadmsync.conf`, and displays a list of the files that differ. For each file name displayed to the console, the difference that exists between the two copies of the same file is also given.

EXAMPLE 7 To Accept the Differences Between Nonreplicated Files

If you decide that the differences between the nonreplicated files displayed by `nhadm synccheck` are not detrimental to your cluster, update `nhadmsync.conf` with the differences by typing:

```
# nhadm syncgen
```

If the *nodeid* of the master node and vice-master node were not entered in the `nhadmsync.conf` file by a previously run `nhadm syncgen` command, a `NODEID` parameter is generated at the top of the file using the `nodeid` of the master and the vice-master nodes in the following format.

`NODEID=`*node1 node2*

Where *node1* and *node2* are the *nodeids* of the master and vice-master nodes, respectively, when `nhadm syncgen` is first run.

The *nodeid* can be preceded by a blank line or a comment. This *nodeid* defines the order of the comparison performed by `synccheck`. The order will remain the same even if a switchover or failover occurs.

Any differences displayed on the console are written to the `nhadmsync.conf` file. The differences for a specific file are printed under the entry for that file in the following format:

```
 =BEGIN
 ...
  =END
```

For example, if the `nhadmsync.conf` file contained the following files:

```
FILE=/etc/ethers
FILE=/etc/hosts
FILE=/etc/netmasks
```

After a `syncgen` the `nhadmsync.conf` file might contain the following information:

```
NODEID=10 20
FILE=/etc/ethers
FILE=/etc/hosts
=BEGIN
5c5,6
```

EXAMPLE 7 To Accept the Differences Between Nonreplicated Files *(Continued)*

```
< 10.250.1.10   MEN-C250-N10    loghost
---
> 10.250.1.20   MEN-C250-N20    loghost
> 10.250.1.10 MEN-C250-N10
8d8
< 10.250.1.20 MEN-C250-N20
=END
FILE=/etc/netmasks
```

The differences printed to the `nhadmsync.conf` file are the differences that would be found by running `diff -b` on the files listed in `nhadmsync.conf`. For more information on the `diff` command, see the `diff`(1) man page.

For further information on the `nhadmsync.conf` file, see the `nhadmsync.conf`(4) man page.

EXAMPLE 8 To Check for New Differences Between Nonreplicated Files

Log in to one of the master-eligible nodes and type:

```
# nhadm synccheck
```

For nonreplicated files listed in `nhadmsync.conf`, any differences that are not already stored in `nhadmsync.conf` are displayed to the console.

EXAMPLE 9 To Change the Location of the File Listing the Nonreplicated Files Compared by `nhadm synccheck`

The `nhadmsync.conf` file is shared by the master and vice-master nodes. To change the name or location of the `nhadmsync.conf` file, use the `-y` option as follows:

```
# nhadm -y pathname syncgen
```

Copying Local Files

This section provides an example for using the `nhadm copy` command to copy local files from the master node to the vice-master node.

```
# nhadm [-d data-file ] copy [ file ]
```

where:

data-file	a list of files to be copied; one file per line
file	additional files to be copied, for example files that you want to copy once only. You can include the files to be copied repeatedly in the *data-file* file.

The `/.rhosts` files must be correctly configured on both nodes to enable remote access via `rcp`.

EXAMPLE 10 To Copy Local DHCP Configuration Files

This example shows how to copy the local DHCP configuration file, `dhcp.dat` from the master node to the vice-master node. The contents of a DHCP configuration file such as `dhcp.dat` could be as follows:

```
/var/dhcp/SUNWrbs1_10_1_1_0
/var/dhcp/SUNWrbs1_10_1_2_0
/var/dhcp/SUNWrbs1_dhcptab
```

To copy `dhcp.dat` from the master node to the vice-master node, log on to the master node and use the `nhadm` command as follows:

```
# nhadm -d dhcp.dat copy
```

The vice-master node now has a copy of the files listed in the `dhcp.dat` file.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhadm
Interface Stability	Evolving

SEE ALSO

`diff`(1), `nhcmmstat`(1M), `nhadmsync.conf`(4)

NAME

nhcmmd – manage cluster membership

SYNOPSIS

/opt/SUNWcgha/sbin/nhcmmd [-h] [-u *URL*]

DESCRIPTION

The Cluster Membership Manager is implemented by the nhcmmd daemon. There is a nhcmmd daemon on each peer node.

The nhcmmd daemon on the master node has the current view of the cluster configuration and communicates its view to the nhcmmd daemons on the other peer nodes. The nhcmmd daemon on the master node determines which nodes are members of the cluster, assigns the roles and attributes to the nodes, detects the failure of nodes and configures routes for reliable transport.

The nhcmmd daemon on the vice-master node monitors the health of the master node. If the master node fails, the vice-master node is able to take over as the master node.

The nhcmmd daemon on each of the peer nodes do not communicate with each other. Each nhcmmd daemon exports an API to the notify clients of changes to the cluster, and to notify services and applications when the cluster membership or master changes. Notification messages describe the membership change and the *nodeid* of the affected node, making it possible for clients to maintain an accurate view of the peer nodes of the cluster.

For information about the CMM API, see the Intro(3CMM) man page.

OPTIONS

-h	Displays help information.
-u *URL*	You must specify the URL of the nhfs.conf file.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
255	An error occurred.

FILES

cluster_nodes_table
: The file that lists the configured nodes of the cluster and describes the nodes' attributes.

nhfs.conf
: The file that contains configuration and addressing information for the different Foundation Services.

target.conf
: The file that contains the *domainid*, attributes, and election roles for each node in the cluster

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	Sparc
Availability	SUNWnhcmb
Interface Stability	Evolving

SEE ALSO `Intro`(3CMM), `cluster_nodes_table`(4), `nhfs.conf`(4), and `target.conf`(4).

NAME

nhcmmqualif – qualify the current node as master

SYNOPSIS

/opt/SUNWcgha/sbin/nhcmmqualif [-v] [-t *timeout*]

DESCRIPTION

The nhcmmqualif command calls the cmm_member_seizequalif(3CMM) function to qualify the current node as master-eligible and start a new master election. This call is only successful if there is no active master node in the cluster and if the current node is master-eligible. If this call is not successful, a 255 exit code is returned. If this call is successful, this command forces the qualification of the current node so that this node becomes the master node. Use this command when no node is qualified to become master. Note that an unsynchronized node will only be elected as master node if its former role was master.

The nhcmmqualif command can only be called from a master-eligible node. This call has one of two outcomes for this node; either the node becomes master, or it reverts to the qualification level it had prior to the nhcmmqualif call.

If you attempt to call nhcmmqualif from a node that is not master-eligible, the command exits with a 255 exit code. If a master is already running when nhcmmqualif is called, the command exits with a 255 exit code. If nhcmmqualif provokes a change in the status of a peer node, a notification is sent by the CMM API.

OPTIONS

The following options are supported by nhcmmqualif:

-t *timeout*
Wait for a specified period of time for a MASTER ELECTED notification. If a master is elected within this period of time, nhcmmqualif is successful and returns 0. If no master is elected within this period of time, nhcmmqualif fails and returns 255.

-v
Verbose mode.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

255 The current node is not master-eligible, there is already a master in the cluster, or no master was elected within the specified timeout (if -t was used).

EXAMPLES

This section contains examples of how to use the nhcmmqualif command.

EXAMPLE 1 To Force Qualification of a Master-Eligible Node

When there is no current master node in the cluster and no node is qualified to be the master node:

- Log in to a master-eligible node.
- Run:

```
# nhcmmqualif
```

EXAMPLE 1 To Force Qualification of a Master-Eligible Node *(Continued)*

```
# echo $?
0
```

The master-eligible node on which you ran the `nhcmmqualif` command is temporarily qualified as the master node.

If you run `nhcmmqualif` on a master-eligible node in a cluster with a valid master node, the node is not forced to become master and the 255 exit status is produced.

EXAMPLE 2 To Requalify a Node Synchronously

Use the `timeout` option to requalify a node synchronously.

- Log in to one of the master-eligible nodes.
- Run:

  # **nhcmmqualif -t** *timeout*

  This command is synchronous. The option `-t` blocks the node until a `MASTER ELECTED` notification is received or the timeout is reached.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	Sparc
Availability	SUNWnhcmb
Interface Stability	Evolving

SEE ALSO

`cmm_member_seizequalif`(3CMM).

NAME

nhcmmrole – get the role of the current node

SYNOPSIS

```
/opt/SUNWcgha/sbin/nhcmmrole [ -hv ] [ -t timeout ]
```

DESCRIPTION

The `nhcmmrole` command gets the role of the current node.

This executable can be used in scripts. Its exit code represents the current role of the node.

OPTIONS

The `nhcmmrole` command can be used with the following options:

`-h`
Provides information about the use of the command.

`-t` *timeout*
Sets the timeout in seconds for this call. The default timeout is 5 seconds.

`-v`
Verbose. This option displays the role of the node, as follows:

```
nhcmmrole: current role role
```

The value for *role* is one of the following: `MASTER`, `VICEMASTER`, `OUT_OF_CLUSTER`, or `IN_CLUSTER`.

EXIT STATUS

The following exit values are returned by `nhcmmrole`:

255	A failure has occurred or the node is not configured in the cluster node table
0	Out of cluster node
1	Master node
2	Vice-master node
3	In cluster node that is neither master nor vice-master.

EXAMPLES

The following example shows how to run `nhcmmrole` to determine the role of a node.

EXAMPLE 1 Running `nhcmmrole` to determine if a node is the master

- Log in to a node.
- Run:

```
# nhcmmrole

# echo $?
1
```

In this example, the current node is the master node.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	Sparc
Availability	SUNWnhcmb
Interface Stability	Evolving

NAME

nhcmmstat – display information about peer nodes, trigger a switchover, or force the qualification of a master-eligible node

SYNOPSIS

nhcmmstat [-h] [-c *command* [-t]] [-n *nodeid*]

DESCRIPTION

The nhcmmstat tool displays information about a peer node or a group of peer nodes, displays notifications sent by the nhcmmd daemon, and performs operations on the cluster. Use this tool at regular intervals when you are performing tasks that might change the status of a node.

The nhcmmstat tool provides the following information:

- The node ID of a node
- That the cluster configuration files contain coherent information
- The role of a node
- The attributes of a node
- That the master and vice-master disks contain the same shared information

You can use the nhcmmstat tool to modify the state of the cluster as shown in the examples section below.

For information about the role and attributes of a node, see the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

OPTIONS

The following options are supported by nhcmmstat:

-c	You can specify the nhcmmstat command to be executed. The specified command is executed and nhcmmstat exits. If this option is not used, you use the nhcmmstat command in an interactive mode. In this case, you must exit using the exit or quit command to return to the cursor.
-h	Displays help.
-n	You can specify the *nodeid* of the node on which you want to run nhcmmstat. This option is obligatory when using the info or potential commands.
-t	Shows the start and end times.

EXTENDED DESCRIPTION

The following commands can be used with the nhcmmstat tool to get information about a single node:

info	Get information about a node in the cluster. You must provide the *nodeid* of the node.
local	Get the *nodeid* of the current node.
master	Get information about the master node.
mynode	Get information about the current node.
potential	Get information about a node that is in the cluster node table but has the CMM_OUT_OF_CLUSTER role. You must provide the *nodeid* of the node.

When a node has the CMM_OUT_OF_CLUSTER role, the nhcmmstat tool gives meaningless values for the following administrative attributes: CMM_ELIGIBLE_MEMBER, CMM_FLAG_DISQUALIFIED, and CMM_FLAG_SYNCHRO_NEEDED.

`vice` Get information about the vice-master node.

The following commands can be used with the `nhcmmstat` tool to get information about all peer nodes.

`all` Get information about all peer nodes except those with the role CMM_OUT_OF_CLUSTER.

`count` Get a count of the nodes in the cluster.

The following commands can be used with the `nhcmmstat` tool to modify the cluster. You must log in as superuser to use these commands.

`reload` Force a reload of the `cluster_nodes_table` configuration. This command can be run from the master node only. The supported operations are add and remove a node in the `cluster_nodes_table` file, with the node powered off.

To add nodes that are not included in the original cluster definition, you must consider how the cluster was installed. For information about how to add new diskless node or dataless nodes to a cluster, see the *Netra High Availability Suite Foundation Services 2.1 6/03 Cluster Administration Guide*. You cannot add a master-eligible node to the cluster because there is a limit of two master-eligible nodes per cluster.

`so` Force mastership change to the vice-master node. This command can be used on the master node only, and when the vice-master is present and able to take the master role.

`squalif` Force the requalification for the current node to make it start an election. This command can be used on a master-eligible node when no peer node is qualified to be master. This command displays the following warning message:

```
Warning! This asynchronous command might take up to 300 s
to succeed!
```

Note that an unsynchronized master-eligible node can only be elected as the master node if its former role was master.

The following commands can be used to exit from or get help with `nhcmmstat`:

`exit` Exit.

`help` Display help information.

`quit` Exit.

If an `nhcmmstat` command fails, an error is displayed.

Notifications

When `nhcmmstat` is used in interactive mode, the following notifications are emitted by the `nhcmmd` daemon and displayed.

`CMM_INVALID_CLUSTER`	`"[USER CB] INVALID CLUSTER"` is displayed. The cluster is not in a coherent state. No information can be returned by the CMM API until the cluster has been returned to a coherent state.
`CMM_VALID_CLUSTER`	`"[USER CB] VALID CLUSTER"` is displayed. The cluster is in a coherent state.
`CMM_STALE_CLUSTER`	`"[USER CB] STALE CLUSTER"` is displayed. The cluster has not had a master node for the preceding ten seconds.
`CMM_MASTER_DEMOTED`	`"[USER CB] master demoted = %d"` is displayed. %d is replaced by the *nodeid* of the demoted master.
`CMM_MASTER_ELECTED`	`"[USER CB] master elected = %d"` is displayed. %d is replaced by the *nodeid* of the new master. When this notification is received, all available information about the master is displayed.
`CMM_MEMBER_LEFT`	`"[USER CB] member left cluster = %d"` is displayed. %d is replaced by the *nodeid* of the node that has left the cluster.
`CMM_MEMBER_JOIN`	`"[USER CB] new node in cluster = %d"` is displayed. %d is replaced by the *nodeid* of the new node.
`CMM_VICEMASTER_DEMOTED`	`"[USER CB] vice-master demoted = %d"` is displayed. %d is replaced by the *nodeid* of the demoted vice-master node.
`CMM_VICEMASTER_ELECTED`	`"[USER CB] vice-master elected = %d"` is displayed. %d is replaced by the *nodeid* of the new vice-master. When this notification is received, all available information about the vice-master is displayed.

When `nhcmmstat` is used in command line mode the notifications are not displayed.

Node Information Displayed

When information is requested for a node, or when notifications of cluster changes are received, the following information is displayed in the order shown:

`nodeid`	The *nodeid* of the current node, followed by `"[This is the current node]"` when the displayed information concerns the current node.
`domain_id`	The *domainid* of the cluster of the current node.
`name`	The name of the node as specified in the `/SUNWcgha/remote/etc/cluster_nodes_table` file.
`role`	The role of the node in the cluster: `master`, `vice-master`, `in`, or `out`. The `master`, `vice-master` and `out` roles correspond to `CMM_MASTER`, `CMM_VICEMASTER`, and `CMM_OUT_OF_CLUSTER`. A node that does not have the `CMM_OUT_OF_CLUSTER` role, is *in*.
`qualified`	`YES` or `NO` is displayed. If `YES`, the node is qualified to be master. If `NO`, the node is not qualified to be master. This information is relevant for master-eligible nodes only.
`synchro`	`NEEDED` or `READY` is displayed. If `NEEDED`, the master and vice-master shared file systems do not contain the same information. If `READY`, the master and vice-master node file systems contain the same information. This information is relevant for master-eligible nodes only.
`frozen`	`YES` or `NO` is displayed. If `YES`, the node is frozen. When a node is frozen, the master cannot change the role of this node even if events require it. If `NO`, the node is not frozen.
`excluded`	*YES* or *NO* is displayed. If *YES*, the node is excluded from the cluster. An excluded node acts as if it has the `CMM_OUT_OF_CLUSTER` role. If *NO*, the node is not excluded.
`eligible`	*YES* or *NO* is displayed. If *YES*, this node can participate in an election and be elected master if it is sufficiently qualified. If *NO*, this node cannot participate in an election.
`incarn`	The incarnation number of the time that the node was last booted. The value is an integer (number of seconds since 00:00 universal coordinated time Jan 1 1970) and a literal representation of this date. For example, `1005833787 (15/11/2001 - 15:16:27).`
`swload_id`	This string indicates the Foundation Services software version. The string `1` is displayed for the Foundation Services.
`CGTP @`	This is the address of the node of the `cgtp0` interface

USAGE

You must log in as superuser to use the `so`, `reload`, and `squalif` commands of the `nhcmmstat` tool.

EXAMPLES

This section contains examples of how to use the `nhcmmstat` tool:

EXAMPLE 1 To Get Information About the Master Node

- Log in to a peer node.
- Type:

```
# nhcmmstat -c master
```

An output similar to the following is displayed:

```
------------------------------
node_id     = 20
domain_id   = 250
name        = netraMEN2-cgtp0
role        = MASTER
qualified   = YES
synchro.    = READY
frozen      = NO
excluded    = NO
eligible    = YES
incarn.     = 1015949483 (12/03/2002 - 17:11:23)
swload_id   = 1
CGTP @      = 10.250.3.20
------------------------------
```

EXAMPLE 2 To Get the *nodeid* of the Current Node

- Log in to a peer node.

  This node becomes the current node.
- Type:

```
# nhcmmstat -c local
```

An output similar to the following is displayed:

```
Local Node id is 10
```

In this example the *nodeid* of the current node is `10`.

You can also find the *nodeid* of a node by using the `ifconfig` command. The *nodeid* corresponds to the host part of the nodes IP address. For more information, see the `ifconfig`(1M) man page.

EXAMPLE 3 To Get Information About a Specific Node

- Log in to a peer node.
- Type:

```
# nhcmmstat -c info -n nodeid
```

An output similar to the following is displayed:

EXAMPLE 3 To Get Information About a Specific Node *(Continued)*

```
------------------------------
node_id     = nodeid
domain_id   = 250
name        = netraMEN1-cgtp0
role        = VICE-MASTER
qualified   = YES
synchro.    = READY
frozen      = NO
excluded    = NO
eligible    = YES
incarn.     = 1008266390 (13/12/2001 - 18:59:50)
swload_id   = 1
CGTP @      = 10.250.3.10
------------------------------
```

EXAMPLE 4 To Force the Qualification of a Node Asynchronously

- Log in to a master-eligible node.
- Type:

```
# nhcmmstat -c squalif
```

- The `nhcmmstat` and `squalif` tool forces the requalification of the current node to make it the master node. This function can only be successful when there is no active master node in the cluster and the current node is a master-eligible node.

  The `squalif` tool is asynchronous. The tool is not blocked while qualification is taking place.

EXAMPLE 5 To Get Information About All Peer Nodes

- Log in to any peer node.
- Type:

```
# nhcmmstat -c all
```

  Information about each peer node is printed to the console.

EXAMPLE 6 To Trigger a Switchover

- Log in to the master node.
- Type:

```
# nhcmmstat -c so
```

  An output similar to the following is displayed:

```
[USER CB] master elected = 10
------------------------------
node_id     = 10
domain_id   = 250
```

EXAMPLE 6 To Trigger a Switchover *(Continued)*

```
name        = netraMEN1-cgtp0
role        = MASTER
qualified   = YES
synchro.    = NEEDED !!!
frozen      = NO
excluded    = NO
eligible    = YES
incarn.     = 1008266390 (13/12/2001 - 18:59:50)
swload_id   = 1
CGTP @      = 10.250.3.10
------------------------------
[USER CB] vicemaster elected = 20
------------------------------
node_id     = 20   [This is the current node]
domain_id   = 250
name        = netraMEN2-cgtp0
role        = VICE-MASTER
qualified   = YES
synchro.    = NEEDED !!!
frozen      = NO
excluded    = NO
eligible    = YES
incarn.     = 1008266566 (13/12/2001 - 19:02:46)
swload_id   = 1
CGTP @      = 10.250.3.20
------------------------------
```

If there is a vice-master qualified to become master, it is elected master, the master becomes the vice-master, and the disks are synchronized. If there is no potential master, `nhcmmstat` does not perform a switchover.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmb
Interface Stability	Evolving

SEE ALSO

`Intro`(3CMM), `nhcmmd`(1M), `cluster_nodes_table`(4), and `nhinstall`(1M).

NAME

nhcrfsadm – command line tool for Reliable NFS administration

SYNOPSIS

/opt/SUNWcgha/sbin/nhcrfsadm [-hv]

/opt/SUNWcgha/sbin/nhcrfsadm -a [-s *server* **]**

/opt/SUNWcgha/sbin/nhcrfsadm -r [-s *server* **]**

/opt/SUNWcgha/sbin/nhcrfsadm [-s *server* **] -f all**

/opt/SUNWcgha/sbin/nhcrfsadm [-s *server* **] -f** *partition-name*

DESCRIPTION

The `nhcrfsadm` tool is a command line tool for the RNFS supervisory daemon administration.

The `nhcrfsadm` tool performs the following tasks:

- Authorizes or refuses the Reliable NFS to start on the vice-master node when a change of disk has been detected
- Performs a replication of one or all of the replicated partitions

Replication must be authorized to start on the vice-master node in the following circumstances:

- The vice-master node has been stopped because of failure or maintenance
- A vice-master node disk containing a replicated partition has been changed

The demand for authorization ensures that the new vice-master node disk is not corrupted.

While the vice-master node is rebooting, the master node `nhcrfsd` daemon asks the operator whether the `nhcrfsd` daemon on the vice-master is allowed to follow the boot procedure or not. The `nhcrfsd` daemon on the vice-master node does not start until an order is given to the `nhcrfsd` daemon on the master node.

OPTIONS

The following options are supported:

`-a`	Authorize replication to start on the vice-master node. This option can be used on the master node only. Replication will start automatically.
-f *all*	Perform a full replication of all partitions present in the RNFS configuration.
-f *partition-name*	Perform a full replication of the partition specified by *partition-name*. The partition must be specified by its name, for example, `/dev/rdsk/c0t0d0s3`.
`-h`	Display help on options.
`-r`	Refuse permission for replication to startup on the vice-master node. This option can be used on the master node only. The `nhcrfsd` daemon on the vice-master node will stop. The vice-master disk can then be removed, replaced, and rebooted.

`-s` *server*	Specify the server whose `nhcrfsd` is to be administrated. The default server is *localhost*. The specified machine should always be the master node.
`-v`	Display the version and the build date of the tool.

USAGE

You must be logged on as superuser to run the `nhcrfsadm` command.

EXIT STATUS

0	Command succeeded
1	Command failed

ENVIRONMENT VARIABLES

The `nhcrfsadm` tool uses a catalog of messages in compliance with International Language Environments (I18N) standards. The `NLSPATH` variable must contain the pattern: `/opt/SUNWcgha/lib/locale/%L/LC_MESSAGES/%N`

FILES

The following file is generated by the `nhcrfsadm` tool:

`/opt/SUNWcgha/lib/locale/C/LC_MESSAGES/CRFS_ADM.cat`
Default catalog of messages.

EXAMPLES

The following examples demonstrates how `nhcrfsadm` is used.

EXAMPLE 1 To Start Replication

After you have changed a disk:

1. Log in to the master node as superuser.
2. Accept the start of replication by typing:

```
# /opt/SUNWcgha/sbin/nhcrfsadm -a
```

EXAMPLE 2 To Resynchronize the Master Node Disk and Vice-Master Node Disk

1. Log in to the master node as superuser.
2. Resynchronize the master and vice-master disks.

```
# nhcrfsadm -f all
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
Availability	SUNWnhfsa, SUNWnhfsb

SEE ALSO

`nhcrfsd`(1M)

NAME

nhcrfsd – Reliable NFS supervisor daemon

SYNOPSIS

/opt/SUNWcgha/sbin/nhcrfsd [-h] [-v] [-u *URL*]

DESCRIPTION

The nhcrfsd daemon manages the Reliable NFS feature of the Foundation Services. The nhcrfsd daemon is started when the system comes up (at init level 2). By default, the nhcrfsd daemon is monitored by the Daemon Monitor, nhpmd(1M).

OPTIONS

The following options are supported:

-h	Display help on options and exit.
-u *URL*	You must specify the URL of the nhfs.conf file.
-v	Display the version and the build date of the daemon and exit.

EXTENDED DESCRIPTION

You must be logged on as superuser to run the nhcrfsd daemon.

The nhcrfsd daemon uses the nhfs.conf file and the /etc/vfstab file.

The nhfs.conf File

The following parameters must be set in the nhfs.conf file:

- RNFS.Slice
- RNFS.StatdAlternatePath
- RNFS.Share

For more information about these parameters, see the nhfs.conf(4) man page.

The /etc/vfstab File

Entries in vfstab use the syntax described in vfstab(4). For each master-eligible node, the vfstab entry must contain the following:

- Information for mounting Reliable NFS replicated slices so that they can be exported by the NFS server.
- Information for mounting the file systems to be replicated by NFS.

The following paths are used by applications:

- Mount Reliable NFS replicated slices locally (on the master-eligible nodes).

```
/dev/dsk/c0t0d0s3 /dev/rdsk/c0t0d0s3 /export ufs - no logging
/dev/dsk/c0t0d0s4 /dev/rdsk/c0t0d0s4 /SUNWcgha/local ufs - no logging
```

  This example mounts the Reliable NFS replicated slice /dev/rdsk/c0t0d0s3 on /export and /dev/rdsk/c0t0d0s4 on /SUNWcgha/local/.

  The RNFS.Share property allows /export to be exported by NFS from the master node.

- Mount the Reliable NFS replicated slices by NFS (on the master-eligible nodes).

```
10.28.3.1:/SUNWcgha/local/export - /SUNWcgha/remote nfs - no fg,hard,intr,noac
```

This example mounts `/SUNWcgha/local/export` of the master node on the `/SUNWcgha/remote` mount point of the master and vice-master nodes.

The specified mount options must be selected according to the type of mount entry.

You can use the `noac` option if the impact on performance is acceptable. For information about how to enable and disable the `noac` option, see the *Netra High Availability Suite Foundation Services 2.1 6/03 Cluster Administration Guide*.

EXIT STATUS

The exit status for `nhcrfsd` is one of the following:

0 Daemon started successfully

1 Daemon failed to start

ENVIRONMENT VARIABLES

The `nhcrfsd` daemon uses a catalog of messages in compliance with International Language Environments (I18N) standards. The `NLSPATH` variable must contain the pattern: `/opt/SUNWcgha/lib/locale/%L/LC_MESSAGES/%N`

FILES

`/etc/opt/SUNWcgha/nhfs.conf`
Configuration and addressing information for the different Foundation Services. The URL for this file could be `file:///etc/opt/SUNWcgha/nhfs.conf`.

`/opt/SUNWcgha/lib/locale/C/LC_MESSAGES/CRFS.cat`
Default catalog of messages

NOTES

All `nhcrfsd` daemon messages are logged with the Solaris™ `syslog` facility. This facility uses message priorities of the `facility.level` form. The `nhcrfsd` daemon uses the `local0` facility and a level that can be `alert`, `err`, or `info`.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
Availability	SUNWnhfsa, SUNWnhfsb

SEE ALSO

`nhfs.conf`(4), `nhcrfsadm`(1M), `nhpmd`(1M), and `syslog.conf`(4).

NAME

nhenablesync – trigger disk synchronization

SYNOPSIS

/opt/SUNWcgha/sbin/nhenablesync

DESCRIPTION

The `nhenablesync` tool enables you to trigger disk synchronization. By default, disk synchronization starts automatically during the boot sequence. You can delay the start of synchronization by setting the `RNFS.EnableSync` property to `False` in the `nhfs.conf` file. If the start of synchronization has been delayed in this way, trigger synchonization by logging into the master node and executing the `nhenablesync` command as follows:

```
# nhenablesync
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhfsb
Interface Stability	Evolving

SEE ALSO

`cluster_nodes_table`(4), `nhfs.conf`(4)

NAME

nhinstall – Foundation Services installation and configuration tool

SYNOPSIS

/opt/SUNWcgha/sbin/nhinstall -h

/opt/SUNWcgha/sbin/nhinstall -r *directory* **[-l** *logfile* **] [** *stage* **]**

DESCRIPTION

The `nhinstall` tool enables you to install and configure the Solaris operating system and the Foundation Services on all the nodes of your cluster.

Before running the `nhinstall` tool, configure it using the following configuration files:

- The `env_installation.conf` file to define the installation environment. See `env_installation.conf`(4).
- The `cluster_definition.conf` file to define the cluster environment. See `cluster_definition.conf`(4).
- (Optional) The `addon.conf` file specifying additional patches and packages that you want to install. This file is useful for upgrading Foundation Services at a later stage. See `addon.conf`(4).
- (Optional) The `nodeprof.conf` file permits the customization of Solaris installation. See `nodeprof.conf`(4).
- (Optional) The `diskless_nodeprof.conf` file permits the customization of Solaris installation on diskless nodes. See `diskless_nodeprof.conf`(4).

To install the Solaris operating system and the Foundation Services on the cluster, type the following command as a superuser on the installation server:

```
# /opt/SUNWcgha/sbin/nhinstall -r config-file-directory
```

where *config-file-directory* is the directory containing the configuration files.

The `nhinstall` tool also supports a recovery mechanism based on a progress indicator. In case of a failure during installation, you can restart the installation at the point where the failure occurred by running the same command.

OPTIONS

`-h`	Help.	
`-r` *directory*	Path to the directory containing the configuration files.	
`-l` *logfile*	Name of a log file. If you specify a log file, the output is recorded in the file in addition to being displayed in the console. In case of an error, the logfile helps to trace the error and to identify the point at which the installation will restart.	
stages	Specify the action you require:	
	`reset`	Reset the progress indicator to force the next installation to restart from the beginning.
	`clear`	Unshare all the exported directories and remove all temporary files before the next installation. The progress indicator is also

removed so that the next time you run the `nhinstall` command, the installation starts from the beginning.

`add` *nodeID nodeID ...*

Add new diskless nodes to a cluster that is already running. *nodeID* is the ID of the new diskless node as defined in the `cluster_definition.conf` file. Before you run the `add` stage:

- Define the new diskless nodes in the `cluster_definition.conf` file by using the `NODE` parameter. For more information, see the `cluster_definition.conf`(4) man page.

  Caution – Define only the new diskless nodes that are to be added to the cluster. Do not define nodes that do not exist and that you may want to add to the cluster in the future. If you do so, the nhinstall tool will fail during installation.
- Execute the `nhinstall` command with the `reset` or `clear` stage.

EXAMPLES

In the following examples, the configuration files are located in the `/home/nhasconf` directory on the installation server.

EXAMPLE 1 To Run the `nhinstall` Command to Start an Installation

```
# /opt/SUNWcgha/sbin/nhinstall -r /home/nhasconf
```

EXAMPLE 2 To Reset the Progress Indicator to Restart an Installation From the Beginning

```
# /opt/SUNWcgha/sbin/nhinstall -r /home/nhasconf reset
```

EXAMPLE 3 To Clear the Installation Environment by Removing Temporary Files and Unsharing Exported Directories

```
# /opt/SUNWcgha/sbin/nhinstall -r /home/nhasconf clear
```

EXAMPLE 4 To Add a New Diskless Node to An Existing Cluster

```
# /opt/SUNWcgha/sbin/nhinstall -r /home/nhasconf reset
# /opt/SUNWcgha/sbin/nhinstall -r /home/nhasconf add 40
```

Where 40 is the *nodeID* of the new diskless node.

ATTRIBUTES See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhins
Interface Stability	Evolving

SEE ALSO `addon.conf`(4), `cluster_definition.conf`(4), `diskless_nodeprof.conf`(4), `env_installation.conf`(4), `nodeprof.conf`(4)

NAME

nhnsmd – Node State Manager daemon

SYNOPSIS

/opt/SUNWcgha/sbin/nhnsmd [-u *URL*]

DESCRIPTION

On the node holding the master role, a logical address is assigned to an interface that is used to connect to the external network. This address is called the *floating external address*. The Node State Manager (NSM) uses the Cluster Membership Manager (CMM) notifications to determine when a node is promoted to or demoted from being the master node. When the NSM is notified that a node has been promoted to master node, it executes a script to configure a floating external address for one of the node's external interfaces. When the NSM is notified that a node has been demoted from the master node, it executes a script to deconfigure the floating external address.

The NSM can be used for tasks other than address management but this is not its intended function. If you configure the NSM for purposes other than to monitor the floating external address of the master node, you must consider the effect of this mechanism on cluster integrity.

OPTIONS

The nhnsmd daemon takes the following option:

-u *URL* — You must specify the URL of the nhfs.conf file.

EXTENDED DESCRIPTION

Launch the nhnsmd manually as a superuser:

```
# ./nhnsmd -u URL
```

The nhnsmd daemon is started at system boot time after the nhcmmd daemon. The nhnsmd daemon registers to receive the following notifications at cluster startup:

- CMM_MASTER_ELECTED
- CMM_MASTER_DEMOTED
- CMM_VICEMASTER_ELECTED
- CMM_VICEMASTER_DEMOTED

The nhnsmd daemon executes a response to notifications in the order in which it receives them. It does not act upon a notification that does not pertain to the current state of the node for which the notification is received.

The nhnsmd daemon maintains persistent state across failures so that when restarted by the Daemon Monitor it can determine whether it has missed any notifications and can take appropriate action. This persistent state is not maintained across a node reboot.

The scripts executed by the nhnsmd daemon must be executable shell scripts.

The first argument of the scripts is the *action* parameter. The action parameter can have two values, enter state and leave state. You can use the same script for both actions. For an *enter state* action, the script passes the string enter as the first argument. For a *leave state* action, the script is passed the string leave as the first argument.

The second argument of the scripts is the *node role* parameter. The *node role* parameter is passed as a lower-case character string. The *node role* parameter can have two values, master and vice-master.

The scripts executed by the `nhnsmd` daemon should not perform actions that change the startup behavior of the node. For any notification received, the `nhnsmd` daemon executes a script that invokes a node to leave an existing state and then a script that invokes that node to enter a new state.

The script provided with Foundation Services handles address failover if the external master address is present on a separate external interface or on a logical interface on the cluster network.

Scripts used by the `nhnsmd` daemon run as asynchronous processes and do not take account of any changes in the cluster state. When writing your own action scripts for the `nhnsmd` daemon do not write scripts that will take a long time to execute or that depend on cluster behavior. Such scripts should not be used as a way of controlling applications or as a replacement for a management framework.

FILES

`nhfs.conf`
Configuration and addressing information for the different Foundation Services. The URL for this file could be `file:///etc/opt/SUNWcgha/nhfs.conf`

`/opt/SUNWcgha/actions/master`
This directory contains scripts for transitions to and from the master state.

`/opt/SUNWcgha/actions/vicemaster`
This directory contains scripts for transitions to and from the vice-master state.

The script names have the `E`*nnxxxxxx* or `L`*nnxxxxxx* form, where `E` denotes an script to enter a state and `L` denotes a script to leave a state, *nn* is a two-digit numeric code, and *xxxxxx* is an arbitrary string of characters. Any files that do not use this naming scheme will be ignored by `nhnsmd`.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhnsa, SUNWnhnsb
Interface Stability	Evolving

SEE ALSO

`nhfs.conf`(4) and `nhpmd`(1M).

NAME

nhpmd – process monitor daemon

SYNOPSIS

/opt/SUNWcgha/lib/sparcv9/nhpmd

DESCRIPTION

The nhpmd daemon provides the Daemon Monitor service. The nhpmd daemon runs at the multiuser level on all nodes in the cluster. The nhpmd daemon surveys other Foundation Services daemons, many Solaris operating system daemons, and some companion product daemons. If a daemon that provides a critical service fails, the nhpmd daemon detects the failure and triggers a recovery response. The recovery response is specific to the daemon that failed.

The nhpmd daemon operates at a higher priority than the other Foundation Services daemons.

Foundation Services daemons and Solaris operating system daemons are launched by a *startup script*. A *nametag* is assigned to the daemon or group of daemons that is launched by each startup script. In some cases, such as for syslogd, a nametag is assigned to only one daemon. In other cases, such as for nfs_client, a nametag is assigned to a group of daemons. If one of the daemons covered by a nametag fails, the recovery response is performed by the nhpmd daemon on all of the daemons covered by that nametag. If the recovery response is to restart the failed daemon, all of the daemons grouped under that nametag are killed and restarted. For a list of monitored daemons and their associated recovery responses, see MONITORED DAEMONS.

Information about monitored daemons can be collected using the nhpmdadm command, as described in the nhpmdadm(1M) man page.

This man page lists the Foundation Services, Solaris operating system, and companion product daemons that are monitored by the nhpmd daemon and describes the recovery action taken by the nhpmd daemon on the node on which the monitored daemon failed.

EXTENDED DESCRIPTION

Note the following before using the nhpmd daemon:

- The initialization process of the Foundation Services alters the /etc/inittab file by replacing the rc2 and rc3 strings with rc2.HA and rc3.HA strings. Do not modify or overwrite rc2.HA or rc3.HA.
- The nhpmd daemon server is started automatically when the system starts up at init level 2 (multi-user mode).
- The nhpmd daemon is a 64–bit application. It cannot run on a 32–bit kernel.
- Files in the /var/run/SUNWcgha/pmd directory, and the directory itself, must not be removed while the nhpmd daemon is running.
- The only signal to which the nhpmd daemon responds is SIGTERM. Provided that the nhpmd daemon is started by superuser, when the SIGTERM signal is sent, the nhpmd daemon stops all monitoring and exits. Previously monitored processes can now be traced or debugged.
- The script provided as an action program to any nhpmdadm command must not be removed; it must exist when the nhpmd daemon attempts to execute it. If the system is out of main resources (memory or processes), the nhpmd daemon might

not be able to launch or relaunch any executables.

- To avoid collisions with other controlling processes, `truss(1)` does not allow a process to be traced that it detects as being controlled by another process by way of the `/proc` interface. The `nhpmd` daemon uses the `/proc` interface to monitor processes and their descendents, therefore, those processes that are submitted to the `nhpmd` daemon using the `nhpmdadm` tool cannot be traced or debugged.
- When you list the processes that are running on the Foundation Services, you see the Foundation Services daemons. Some of the daemons delivered with the Foundation Services are part of the Foundation Services internal subsystem and cannot be publicly accessed. Some daemons run only on the master and vice-master nodes, and some run on all peer nodes.
- When you list the running processes, the name of the Node Management Agent daemon does not appear as `nma`. To see the process name for the Node Management Agent daemon, use the `ps` command. The Process ID (PID) of this daemon is in `/var/run/SUNWcgha/nma.pid`.

MONITORED DAEMONS

The following lists give the nametag and associated recovery response of the Foundation Services, Solaris operating system, and companion product daemons that are monitored by the `nhpmd` daemon. The recovery responses listed are the default values. You can specify the number of times the nhpmd daemon tries to restart a daemon if you create the `nhpmd.conf` file. For a description of these daemons, see their man pages. For information about the `nhpmd.conf` file, see the `nhpmd.conf`(4) man page.

Monitored Daemons in the Foundation Services

The following list gives the nametag and recovery response of the monitored daemons in the Foundation Services.

Daemon - `nhcrfsd` — Nametag - `nhcrfsd`

Recovery response - relaunches the daemon up to three times. In some cases, the `nhcrfsd` daemon detects a fatal error and reboots the node.

Daemon - `nhcmmd` — `Nametag - nhcmmd`

Recovery response - does not restart the daemon; reboots the node on which it failed.

Daemon - `nhprobed` — Nametag - `cgha_probe`

Recovery response - does not restart the daemon; reboots the node on which it failed

Daemon - `nma` — `Nametag - nma`

Recovery response - relaunches the daemon up to 10 times then exits

Daemon - `nhwdtd` — `Nametag - cgha_nhwdt`

	Recovery response - relaunches the daemon up to three times then reboots the node on which it failed

Monitored Daemons in the Companion Products

The following list gives the nametag and recovery response of the monitored daemons in the companion products.

Daemon - `nskernd`	Nametag - `sndr.nskernd` Recovery response - does not restart the daemon; reboots the node on which it failed
Daemon - `sndrd`	Nametag - `sndr.sndrd` Recovery response - does not restart the daemon; reboots the node on which it failed

Monitored Daemons in the Solaris Operating System

The following list gives the nametag and recovery response of the monitored daemons in the Solaris operating system.

Daemon - `cron`	Nametag - `cron` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemons - `dsvclokd`, `in.dhcpd`	Nametag - `dhcpd` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemons - `fnsypd`, `keyserv, nis_cachemgr`, `rpcbind, rpc.nisd`, `rpc.nispasswdd`, `rpc.yp, ypbind`, `ypserv, ypxfrd`	Nametag - `rpc` Recovery response - sends an error message
Daemons - `inetd`, `in.named`	Nametag - `inetsvc` Recovery response - reboots the node on which it failed
Daemon - `in.routed`	Nametag - `inetinit.routed` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `in.rdisc`	Nametag - `inetinit.rdisc` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails

Daemon - `in.rdisc`	Nametag - `nfs.client` when no `nhcrfsd` daemon is running on the local node Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `lockd`	Nametag - `nfs.client.lockd` when a `nhcrfsd` daemon is running on the local node Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `mountd`, `nfsd`, `nfslogd`	Nametag - `nfs.server` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `nscd`	Nametag - `nscd` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `slpd`	Nametag - `slpd` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `statd`	Nametag - on the master node `nfs.client.statd.crfs` Nametag - on the vice-master node `nfs.client.statd` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `syslogd`	Nametag - `syslog` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `utmpd`	Nametag - `utmpd`

	Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails
Daemon - `xntpd`	Nametag - `xntpd` Recovery response - relaunches the daemon up to two times and logs an error message if the second relaunch fails

DIAGNOSTICS

Diagnostic messages are logged to the console or in a file, depending on the system's `syslog` `local0` facility settings.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
Availability	SUNWnhpma, SUNWnhpmb, SUNWnhpms

SEE ALSO

`nhpmdadm`(1M), `nhpmd.conf`(4)

NAME

nhpmdadm – process monitor daemon administration tool

SYNOPSIS

/opt/SUNWcgha/sbin/nhpmdadm [-l *nametag* **] [-L]**

DESCRIPTION

The nhpmdadm tool provides the administrative command-line interface to the process monitor daemon, nhpmd(1M). See the nhpmd man page for more information about this daemon.

The options with nhpmdadm that are available are -l and -L.

Note – Do not kill or stop nhpmdadm -L or nhpmdadm -l *nametag* commands.

OPTIONS

The options -l and -L are supported for troubleshooting purposes. They can be used to get information about monitored processes.

-l *nametag* — Prints out status information about nametag. The output from this command is useful mainly for diagnostic purposes.

-L — Returns a list of all tags currently running that belong to the user that issued the command or, if the user is superuser, all tags running on the server.

For a list of the *nametags* and the daemons to which they correspond, see the nhpmd(1M) man page.

EXAMPLES

This section provides examples of using the nhpmdadm command.

EXAMPLE 1 How to use nhpmdadm

- To get all nametags:

```
# nhpmdadm -L
```

Result:

```
tags: utmpd sendmail nscd cron syslog nfs.client inetsvc rpc
```

- To get detailed information about a specific nametag, for example, cron:

```
# nhpmdadm -l cron
```

Result:

```
# nhpmdadm -c cron -n 2 -a /etc/opt/SUNWcgha/init.d/cron.HA.fail
        environment:
                PATH=/usr/sbin:/usr/bin
                TZ=MET
                    ...
                _
        retries: 0
        owner: root
        monitor children: all
        pids: 341
```

EXAMPLE 2 To Verify a Daemon Is Being Monitored

- Log in to a node, as superuser.
- Select a daemon to investigate.
- Confirm that the daemon is running.

  ```
  $ pgrep -x daemon-name
  ```

- Note the process ID for the daemon.
- Find the *nametag* for the daemon:

  ```
  # /opt/SUNWcgha/sbin/nhpmdadm -L
  ```

  Alternatively, use the tables in the `nhadm`(1M) man page to find the Daemon Monitor *nametag* that corresponds to the daemon that you want to investigate.
- Using the daemon `nametag`, run:

  ```
  # /opt/SUNWcgha/sbin/nhpmdadm -l nametag
  ```

  A list of process IDs is displayed for the Daemon Monitor *nametag*.
- Confirm that the process ID entry for this daemon in the list is the same as the process ID returned by the `pgrep` command.

  If this is the case, the daemon is being monitored. If not, the daemon is not being monitored.

EXIT STATUS

0	The command was completed successfully.
1	`nametag` doesn't exist, or there was an attempt to create a `nametag` that already exists.
2	The command timed out.
>2	An error occurred.

MESSAGE LISTS

Log file outputs from the `nhpmdadm` tool are made through `stderr`.

The log files from the Foundation Services daemons are internationalized using the `SOLARIS LC_MESSAGES` database. The message file for the process daemon monitor is in `/opt/SUNWcgha/lib/locale/C/LC_MESSAGES/nhpmd.mo`

The following is a list of messages that are output to the log files by the `nhpmdadm` daemon.

- `<tagname> No such <nametag> registered`

  The specified nametag is not recognized.
- `Missing command argument`

  The `nametag` argument is missing from the `-l` option.
- `Too many command line arguments`

  There are too many options specified in the command line.

CAUTION The `nhpmdadm` command is a 64–bit application, and cannot be run on a 32–bit kernel.

ATTRIBUTES See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
Availability	SUNWnhpma, SUNWnhpmb, SUNWnhpms

SEE ALSO `nhpmd`(1M)

NAME

nhpmdadmwrapper – configure important values, like retry-count

SYNOPSIS

/opt/SUNWcgha/sbin/nhpmdadmwrapper

DESCRIPTION

The `nhpmdadmwrapper` command is exclusively used in HA start/stop scripts where you need to configure important values such as *retry-count* for a daemon (the number of times a daemon is restarted before calling the failure script).

By using `nhpmdadmwrapper` in HA scripts, *retry-count* no longer needs to be hardwired in these script but can be initialized in a dedicated configuration file, `/etc/opt/SUNWcgha/nhpmd.conf`.

The value specified in the script is still mandatory, and is considered the default value for *retry-count*. If, however, a value is specified in this configuration file, the value takes precedence over the default value.

The configuration file, `/etc/opt/SUNWcgha/nhpmd.conf`, contains lines like :

```
<nametag>_RetryCount = <value>
```

For the `statd` daemon:

```
nfs.client.statd.crfs_RetryCount = 2
```

Note that:

- You must insert spaces or tabs before and after "=".
- The `-n` option should be present in the HA scripts for the daemon start command so that the configuration mechanism works (the `-n` option will not be added if it is not present).

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
Availability	SUNWnhpma, SUNWnhpmb, SUNWnhpms

SEE ALSO

`nhpmdadm`(1M), `nhpmd.conf`(4)

NAME

nhprobed – test accessibility of physical interfaces

SYNOPSIS

/opt/SUNWcgha/sbin/nhprobed [-h] [-u *URL*]

DESCRIPTION

Each peer node runs a daemon, `nhprobed`, that periodically sends a heartbeat in the form of an IP packet. The `nhprobed` daemon sends the heartbeats by multicast.

Heartbeats are sent through each of the two physical interfaces of each peer node. When a heartbeat is detected through a physical interface, it indicates that the node is reachable and that the physical interface is alive. If a heartbeat is not detected for a period of time exceeding the detection delay, the node or one of its physical interfaces is considered to have failed. If both of the node's physical interfaces fail, the node itself is considered to have failed. Heartbeats are broadcast at the rate of 3 per 900 milliseconds, and at least one heartbeat must be detected each 900 milliseconds.

On the master-eligible nodes, the `nhprobed` daemon receives a list of nodes from the `nhcmmd` daemon. The `nhprobed` daemon monitors the heartbeats of the nodes on the list. On the master node, the list contains all of the master-ineligible nodes and the vice-master node. On the vice-master node, the list contains the master node only.

On the master-eligible nodes, the `nhprobed` daemon notifies the `nhcmmd` daemon when, for any node on its list, any of the following events occur:

- One link becomes available, indicating that the node is accessible through the link.
- One link becomes unavailable, indicating that the node is not accessible through the link.
- The node becomes available, indicating that the first link to the node becomes available.
- The node becomes unavailable, indicating that the last available link to the node becomes unavailable.

EXTENDED DESCRIPTION

The `nhprobed` daemon uses a kernel module, `hbmod`, and a kernel driver, `hbdrv`. The kernel module and kernel driver manage the send and receive mechanisms of the `nhprobed` daemon. Because the kernel module and kernel driver operate in the kernel space, the `nhprobed` daemon is more robust against system overload.

Note – The probe heartbeat of the Foundation Services 2.1 is incompatible with that of the Foundation Services 2.1 6/03. All nodes in a cluster must have the same probe installation. For information about how to install the packages for the `nhprobed` daemon, the kernel module and kernel driver, see the *Netra High Availability Suite Foundation Services 2.1 6/03 Custom Installation Guide*.

The `nhcmmd` daemon and `nhprobed` daemon communicate through an Internet socket, AF_INET, in connection oriented mode. The socket port number can be configured in the `/etc/services` file, using the service name `cmm_cgtp_probe`.

OPTIONS

The `nhprobed` daemon takes the following options:

`-h`	Displays help information.
`-u` *URL*	You must specify the URL of the `nhfs.conf` file.

For more information, see the `nhfs.conf`(4) man page.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

FILES

`URL` Common configuration file. The file that contains configuration and addressing information for the individual Foundation Services.

The URL for this file could for example be `file:///etc/opt/SUNWcgha/nhfs.conf`

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmb, SUNWnhhb
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `nhcmmd`(1M), and `nhfs.conf`(4)

NAME

nhsched – Display the scheduling parameters of the Foundation Services processes

SYNOPSIS

/opt/SUNWcgha/sbin/nhsched -a

/opt/SUNWcgha/sbin/nhsched -h

/opt/SUNWcgha/sbin/nhsched -i *pid* | *process-name*

/opt/SUNWcgha/sbin/nhsched -u *URL*

DESCRIPTION

The `nhsched` command displays the scheduling parameters of the Foundation Services processes.

OPTIONS

The `nhsched` command can be used with the following parameters:

`-a`
Display the current scheduling base priority configuration.

`-i` *pid* | *process-name*
Display the scheduling parameters for the specified process.

`-h`
Display help information.

`-u` *URL*
Specify an alternative URL for the `nhfs.conf` file. If you do not use this option, the URL `file:///etc/opt/SUNWcgha/nhfs.conf` is used.

EXIT STATUS

The exit status for the `nhsched` command is one of the following:

0 Success

1 Error

EXAMPLES

This section gives examples of how to use the `nhsched` command.

EXAMPLE 1 Get Information About the Current Configuration

```
# /opt/SUNWcgha/sbin/nhsched -a
Current base priority for FIFO        class : 10 (min=0, max=20)
Current base priority for RR            class : 10 (min=0, max=20)
Current base priority for OTHER        class : 10 (min=0, max=20)
```

The maximum and minimum values are retrieved from system information.

EXAMPLE 2 Display the Scheduling Parameters for a Process

Display the scheduling parameters for the `nhpmd` process:

```
# /opt/SUNWcgha/sbin/nhsched -i nhpmd
process id           : 62
process name         : nhpmd
scheduling policy    : RR
scheduling priority  : 40
time quantum         : 0s:100000000ns
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcdt
Interface Stability	Evolving

SEE ALSO

`nhfs.conf`(4)

NAME

nhsmctsetup – create the SMCT environment

SYNOPSIS

/opt/SUNWcgha/nhsmct/bin/nhsmctsetup [-h]

/opt/SUNWcgha/nhsmct/bin/nhsmctsetup -w *smct-dir* **-s** *solaris-dist* **[-h]**

/opt/SUNWcgha/nhsmct/bin/nhsmctsetup -d -w *smct-dir* **-s** *solaris-dist* **[-h]**

/opt/SUNWcgha/nhsmct/bin/nhsmctsetup -w *smct-dir* **-s** *solaris-dist* **-r** *nhas-pkgs* **[-p** *nhas-patch***] [-h]**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `nhsmctsetup` command creates the SMCT environment and directories on the build server or the installation server if it is configured as a build server.

An SMCT environment must be created for each user planning to deploy the Foundation Services software. Use the Korn shell when creating the SMCT environment.

You can run `nhsmctsetup` as a single command with the options described in the following section. You must include the `-r` option to create an environment that runs both `sl`*xxx* and `fl`*xxx* commands. If you do not specify this option, your environment can only run the `fl`*xxx* commands.

To run `nhsmctsetup` in an interactive mode, do not specify any options.

The `nhsmctsetup` command must be run as a superuser.

OPTIONS

The options that you can use with the `nhsmctsetup` command are:

`-w` *smct-dir*	Directory where the SMCT environment will be created.
`-s` *solaris-dist*	Directory where the Solaris JumpStart distribution is located.
`-r` *nhas-pkgs*	Directory where the Foundation Services runtime packages are located. This option is necessary if you want to run the software load commands (`sl`*xxx*).
`-d`	Reconfigures the SMCT environment with regard to the new SMCT directory and Solaris distribution, after it has been copied to a new location.
`-p` *nhas-patch*	Directory where the Foundation Services patches are located. This option is necessary if you want to run the software load (`sl`*xxx*) commands.
`-h`	Displays help information.

PARAMETERS

The `nhsmctsetup` command creates an environment definition file, `smct.env`, that is stored in the *smct-dir*`/scripts` directory. Do not manually modify this file.

This file contains the following environment variables:

`NHAS_PKG_DIR`
Foundation Services runtime packages repository

`NHAS_PAT_DIR`
Foundation Services runtime patches repository

`NHAS_PROD_DIR`
Foundation Services product installation directory

`SMCT_SOL_DIR`
Location of the Solaris JumpStart distribution

`SMCT_VER`
SMCT version

`SMCT_ENV_DIR`
SMCT environment root directory

`SMCT_ETC_DIR`
SMCT default root configuration directory

`SMCT_MODELS_DIR`
SMCT default model templates directory

`SMCT_HARDWARE_DIR`
SMCT default hardware configuration files directory.

`SMCT_SERVICES_DIR`
SMCT default Foundation Services configuration files directory

`SMCT_DEFAULT_CONFIG_DIR`
SMCT default configuration files directory

`SMCT_SWLREP_DIR`
SMCT software load repository

`SMCT_SOFTREP_DIR`
SMCT software repository

Create aliases in your environment to the `sl`*xxx* and `fl`*xxx* commands by running the following command in the Korn shell:

```
# . $SMCT_ENV_DIR/scripts/smct.env
```

This command must be run each time you return to your SMCT environment.

To check the values that are currently configured for the SMCT environment variables, use the following command:

```
# env | grep SMCT
```

ATTRIBUTES See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO `flconfig`(1M), `flcreate`(1M), `fldeploy`(1M), `flinstall`(1M), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M), `slexport`(1M)

NAME

nhwdtd – Watchdog Timer daemon

SYNOPSIS

/opt/SUNWcgha/sbin/nhwdtd [-u *URL* **]**

DESCRIPTION

The `nhwdtd` daemon implements the Watchdog Timer service.

The `nhwdtd` daemon monitors the hardware watchdog on the nodes of the cluster, at the lights-off management (LOM) level. If you are using CompactPCI servers, do not use the Watchdog Timer service provided by the Foundation Services. The CompactPCI hardware watchdogs operate at the OpenBoot PROM level and are monitored by the platform's software.

On hardware watchdogs that operate at the LOM level, the `nhwdtd` daemon monitors a node for operating system hang, but does not monitor the boot process.

The `nhwdtd` daemon can be configured differently on each node depending on your requirements. For information on how to configure the `nhwdtd` daemon, see the `nhfs.conf`(4) man page.

The `nhwdtd` daemon is monitored by the `nhpmd` daemon. If the `nhwdtd` fails, it is relaunched three times. If the `nhwdtd` daemon fails a third time, the node is rebooted.

The `nhwdtd` daemon operates at a lower priority than the `nhpmd` daemon, but at a higher priority than the other Foundation Services daemons.

OPTIONS

The `-u` option enables you to specify the URL of the `nhfs.conf` file. For example:

```
% /opt/SUNWcgha/sbin/nhwdtd -u file:///etc/opt/SUNWcgha/nhfs.conf
```

The URL `file:///etc/opt/SUNWcgha/nhfs.conf` is the default value.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhwdt
Interface Stability	Evolving

SEE ALSO

`nhpmd`(1M), `nhfs.conf`(4)

NAME

NMA – Node Management Agent daemon

SYNOPSIS

```
/etc/opt/SUNWcgha/init.d/nma [ start | stop ]
```

DESCRIPTION

The Node Management Agent (NMA) is a Java Management eXtensions (JMX) compliant management agent based on the Java Dynamic Management Kit (DMK). The NMA provides access to cluster statistics and operations through:

- SNMP
- RMI
- HTML over HTTP

OPTIONS

The following options are supported:

`-start` Start the `NMA`

`-stop` Stop the `NMA`

USAGE

Configure the NMA by editing the Java property file, *installDir*`/etc/opt/SUNWcgha/nma.properties`.

Connector parameters control which Java DMK connector MBeans are instantiated, and which port the connector uses. Edit the `nma.properties` file to specify the port. Do not use the standard RMI port, `1099` because it can cause communication breakdown if an RMI registry or Java DMK agent is running on that port while the NMA is running.

FILES

installDir`/etc/opt/SUNWcgha/nma.properties`
Properties file

installDir`/etc/opt/SUNWcgha/nma.security`
SNMP security configuration file

installDir`/etc/opt/SUNWcgha/nma.notifs.txt`
Notification types and the target to which each type will be sent

installDir`/etc/opt/SUNWcgha/nma.params.txt`
Communication, implementation and accessibility configuration

installDir`/etc/opt/SUNWcgha/nma.targets.txt`
Targets to which the NMA sends SNMP traps

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
Availability	SUNWnhmaj, SUNWnhmal, SUNWnhmas, SUNWnhmad

SEE ALSO

`nma.notifs.txt`(4), `nma.params.txt`(4), `nma.properties`(4), `nma.security`(4), `nma.targets.txt`(4)

NAME

slconfig – SMCT command to add user defined configuration data to the software load

SYNOPSIS

`slconfig -n` *swl-name* **`-v`** *swl-version* **`[-c`** *config-dir*`]` **`[-l`** *logfile*`]` **`[-V`** *verbosity-level*`]` **`[-h]`**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `slconfig` command adds user defined configuration data to the software load repository.

Configure the files for user applications before running the `slconfig` command. The configuration files for user applications describe the user-defined configuration data that is required for each node group. For more information on these files, see the `userapp.conf`(4) man page.

OPTIONS

The options that you can use with the `slconfig` command are:

`-n` *swl-name*	Name of the software load to be configured. This is an ASCII string.
`-v` *swl-version*	Version of the software load to be configured. This is an ASCII string.
`-c` *config-dir*	Directory where the configuration files are located. This option overrides the configuration directory specified by the SMCT variable `SMCT_DEFAULT_CONFIG_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-l` *log-file*	Name of the file that is sent information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`flconfig`(1M), `flcreate`(1M), `fldeploy`(1M), `flinstall`(1M), `nhsmctsetup`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M), `slexport`(1M), `userapp.conf`(4)

NAME

slcreate – SMCT command to prepare the data for a generic flash archive.

SYNOPSIS

slcreate -n *swl-name* **-v** *swl-version* **[-f] [-c** *config-dir***] [-l** *logfile***] [-V** *verbosity-level***] [-h]**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `slcreate` command prepares the data needed to create a generic flash archive.

The `slcreate` command performs the following functions:

- Copies the Foundation Services packages and any user application packages into the software repository.
- Creates a software load in the software load repository. The Solaris JumpStart files are generated for master-eligible and dataless node groups.

  If the software load that is being created already exists, you are prompted to confirm that the current software load should be overwritten or that the operation should be cancelled. If `slcreate` is run with the `-f` option, this prompt does not appear.

The `slcreate` command is run once for each software load because all node groups are processed in one operation.

Data is created from information in several configuration files. Configure the following files before running the `slcreate` command:

- `cluster.conf`

  The `cluster.conf` file contains a logical view of the cluster in terms of nodes, node groups, and domains. For more information see the `cluster.conf`(4) man page.
- `machine.conf`

  The `machine.conf` file describes the cluster in terms of hardware elements, disk layout, and file system. For more information see the `machine.conf`(4) man page.

Optionally, configure the following files before running the `slcreate` command:

- `network.conf`

  The `network.conf` file describes the network parameters for the target cluster. For more information see the `network.conf`(4) man page.
- Solaris JumpStart `profile` file

  Each Solaris JumpStart `profile` file defines the set of Solaris operating environment packages to be installed on each master-eligible and dataless node group. There is one `profile` file for each master-eligible or dataless node group. A `profile` file, named `profile.proto`, is provided in the `/opt/SUNWcgha/nhsmct/etc/jumpstart/`*Solaris_version* directory. If you want to modify this file, copy this file to the `SMCT_DEFAULT_CONFIG_DIR/jumpstart/`*Solaris_version* directory with the file

name `profile.proto.`*node-group-name*.. Alternatively, you can copy it to the *config-dir*`/jumpstart/`*Solaris_version*`/profile.proto.`*node-group-name* where *config-dir* is specified by the `-c` option, which is described in the following section.

- Solaris JumpStart `sysidcfg` file

  The Solaris JumpStart system identification configuration file contains information for the target cluster such as name service, timezone and superuser password. There is one `sysidcfg` file for each master-eligible and dataless node group. A `sysidcfg` file template, named `sysidcfg.proto`, is provided in the `/opt/SUNWcgha/nhsmct/etc/jumpstart/` directory. If you want to modify this file, copy this file to the `SMCT_DEFAULT_CONFIG_DIR/jumpstart/` directory with the file name `sysidcfg.proto` or `sysidcfg.proto.`*node-group-name*. Alternatively, you can copy it to *config-dir*`/jumpstart/sysidcfg.proto` or *config-dir*`/jumpstart/sysidcfg.proto.`*node-group-name* where *config-dir* is specified by the `-c` option, which is described in the following section.

- Solaris JumpStart `begin` file

  There is one `begin` file for each master-eligible and dataless node group. A `begin` file template, named `begin.proto`, is provided in the `/opt/SUNWcgha/nhsmct/etc/jumpstart/` directory. If you want to modify this file, copy this file to the `SMCT_DEFAULT_CONFIG_DIR/jumpstart/` directory with the file name `begin.proto` or `begin.proto.`*node-group-name*. Alternatively, you can copy it to *config-dir*`/jumpstart/begin.proto` or *config-dir*`/jumpstart/begin.proto.`*node-group-name* where *config-dir* is specified by the `-c` option, which is described in the following section.

- Software configuration files

  Each software configuration file defines additional software and patches that are to be installed for each node group. There may be a software configuration file for each node group. For more information on software configuration files, see the `software.conf`(4) man page.

OPTIONS

The options that you can use with the `slcreate` command are:

`-n` *swl-name*	Name of the software load to be created. This is an ASCII string.
`-v` *swl-version*	Version of the software load to be created. This is an ASCII string.
`-c` *config-dir*	Directory where the configuration files are located. This option overrides the configuration directory specified by the SMCT variable `SMCT_DEFAULT_CONFIG_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-f`	Removes the software load identified by *swl-name* and *swl-version* if it already exists.

`-l` *log-file*	Name of the file that is sent information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`cluster.conf`(4), `flconfig`(1M), `flcreate`(1M), `fldeploy`(1M), `flinstall`(1M), `machine.conf`(4), `nhsmctsetup`(1M), `network.conf`(4), `slconfig`(1M), `sldelete`(1M), `sldeploy`(1M), `slexport`(1M), `software.conf`(4)

NAME

sldelete – SMCT command to delete a software load

SYNOPSIS

`sldelete -n` *swl-name* **`-v`** *swl-version* **`[-f] [ -l`** *logfile*`]` **`[-V`** *verbosity-level*`]` **`[-h]`**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `sldelete` command removes a software load from the SMCT environment by deleting the software load data from the software load repository. By default, the related software distributions are also deleted if they do not belong to other software loads.

You can remove all software in a software load by using the `-f` option. This option must be used carefully as it can create discrepancies with the software load stored in the software load environment.

OPTIONS

The options that you can use with the `sldelete` command are:

`-n` *swl-name*	Name of the software load to be removed. This is an ASCII string.
`-v` *swl-version*	Version of the software load to be removed. This is an ASCII string.
`-f`	Forces the removal of the software load identified by *swl-name* and *swl-version* even if this software belongs to other software loads.
`-l` *log-file*	Name of the file that is sent information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`flconfig`(1M), `flcreate`(1M), `fldeploy`(1M), `flinstall`(1M), `nhsmctsetup`(1M), `slconfig`(1M), `slcreate`(1M), `sldeploy`(1M), `slexport`(1M)

NAME

sldeploy – SMCT command to generate the Foundation Services and Solaris operating system configuration files for a software load

SYNOPSIS

`sldeploy -n` *swl-name* **`-v`** *swl-version* **`[-c`** *config-dir*`]` **`[-l`** *logfile*`]` **`[-V`** *verbosity-level*`]` **`[-h]`**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `sldeploy` command generates the Foundation Services and Solaris operating system configuration files associated to a software load.

The configuration files generated for the Foundation Services are used to configure services such as Reliable NFS and the Cluster Membership Manager (CMM). The `sldeploy` command also generates the `nhfs.conf` file.

Configure the following files before running the `sldeploy` command:

- `cluster.conf`

  The `cluster.conf` file contains a logical view of the cluster in terms of nodes, node groups, and domains. For more information see the `cluster.conf`(4) man page.

- `machine.conf`

  The `machine.conf` file describes the cluster in terms of hardware elements, disk layout, and file system. For more information see the `machine.conf`(4) man page.

- `network.conf`

  The `network.conf` file describes the network parameters for the target cluster. For more information see the `network.conf`(4) man page.

Optionally, configure the following files before running the `sldeploy` command:

- Solaris JumpStart `sysidcfg` file

  The Solaris JumpStart system identification configuration file contains information for the target cluster such as name service, timezone and superuser password. There is one `sysidcfg` file for each master-eligible and dataless node group. A `sysidcfg` file template, named `sysidcfg.cluster`, is provided in the `/opt/SUNWcgha/nhsmct/etc/jumpstart` directory. It must be copied to the `SMCT_DEFAULT_CONFIG_DIR/jumpstart` directory with the file name `sysidcfg.cluster` or `sysidcfg.cluster.`*node-group-name*.

- Solaris JumpStart `begin` file

  There is one `begin` file for each master-eligible and dataless node group. A `begin` file template, named `begin.cluster`, is provided in `/opt/SUNWcgha/nhsmct/etc/jumpstart`. If you want to modify this file, copy this file to the `SMCT_DEFAULT_CONFIG_DIR/jumpstart` directory with the file name `begin.cluster` or `begin.cluster.`*node-group-name*. Alternatively, you can copy this file to *config-dir*`/jumpstart/begin.cluster` or *config-dir*`/jumpstart/begin.cluster.`*node-group-name* where *config-dir* is specified by the `-c` option, which is described in the following section.

OPTIONS

The options that you can use with the `sldeploy` command are:

`-n` *swl-name*	Name of the software load to be deployed. This is an ASCII string.
`-v` *swl-version*	Version of the software load to be deployed. This is an ASCII string.
`-c` *config-dir*	Directory where the configuration files are located. This option overrides the configuration directory specified by the SMCT variable `SMCT_DEFAULT_CONFIG_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-l` *log-file*	Name of the file that is sent information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`cluster.conf`(4), `flconfig`(1M), `flcreate`(1M), `fldeploy`(1M), `flinstall`(1M), `machine.conf`(4), `network.conf`(4), `nhsmctsetup`(1M), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `slexport`(1M), `nhfs.conf`(4)

NAME

slexport – SMCT command to copy software load data to an export directory

SYNOPSIS

slexport -n *swl-name* **-v** *swl-version* **[-e** *export-dir***] [-r] [-f] [-l** *logfile***] [-V** *verbosity-level***] [-h]**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `slexport` command copies software load data to an export directory. The export directory interfaces directly with the SMCT flash archive commands (fl*xxx*).

The `slexport` command copies the software load output from the commands `slcreate`, `slconfig`, and `sldeploy` to a specified directory.

The `slexport` is run once at each stage of the configuration process and all node groups are processed each time it is executed.

OPTIONS

The options that you can use with the `slexport` command are:

`-n` *swl-name*	Name of the software load to be exported. This is expressed as an ASCII string.
`-v` *swl-version*	Version of the software load to be exported. This is expressed as an ASCII string.
`-e` *export-dir*	Directory where the software load data will be exported to. The use of this option enables you to use the flash archive commands from another server. The default directory is specified by the SMCT variable `SMCT_EXPORT_DIR`, which is defined in the `smct.env` file. For more information on the `smct.env` file, see the `nhsmctsetup`(1M) man page.
`-r`	Copy the software associated with the software load into the export directory. This option is required if the generic flash archive is generated on a machine other than the build server, for example the installation server.
`-f`	Force the removal of the exported data if such data exists in the export directory. By default, you are prompted to choose whether the data is to be kept or deleted.
`-l` *log-file*	Name of the file that is sent information and error messages. By default these messages are displayed on the console.
`-V` *verbose-level*	Verbosity level. Values are 1, 2, or 3, where 1 is minimal traces and 3 is detailed information.
`-h`	Displays help information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`flconfig`(1M), `flcreate`(1M), `fldeploy`(1M), `flinstall`(1M), `nhsmctsetup`(1M), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M)

Netra HA Suite CMM Library Functions

NAME

cmm_cmc_filter – define notification filtering

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_cmc_filter(cmm_cmcfilter_t const action,
     cmm_cmchanges_t const * const notifications_list, uint32_t const
     notifications_count);
```

DESCRIPTION

The `cmm_cmc_filter()` function defines the list of notifications sent to an application that is registered to receive Cluster Membership Manager notifications. An application registers to receive notifications by calling the `cmm_cmc_register`(3CMM) function. By default, when an application calls this function, the application receives a notification for every change that occurs in the cluster state. An application can operate when viewing a subset of the notifications. By using the `cmm_cmc_filter()` function, an application defines the list of notifications that it receives.

Note – For information on the notification sequences and the various scenarios, see the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_cmc_filter()` function takes the following parameters:

action	Specifies the action to be performed on the current filter (add, remove, set).
notifications_list	An array of `cmm_cmchanges_t`. This array represents the modifications to be applied to the current filter.
notifications_count	Specifies the number of elements in *notifications_list*.

- The *action* parameter is set to:

`CMM_CMC_NOTIFY_REM`	To remove some notifications from the set currently in the filter.
`CMM_CMC_NOTIFY_ADD`	To receive a given set of notifications in addition to the set currently in the filter.
`CMM_CMC_NOTIFY_SET`	To define a completely new set of notifications.
`CMM_CMC_NOTIFY_ALL`	To receive all notifications.
`CMM_CMC_NOTIFY_NONE`	To receive no notifications. This is not a way to remove a registration. Use `cmm_cmc_unregister()` to stop `nhcmmd` sending notifications.

- The *notifications_list* parameter is the set of notifications. In the case of `CMM_CMC_NOTIFY_ALL` and `CMM_CMC_NOTIFY_NONE`, this argument is ignored and not tested.

- The *notifications_count* parameter is the number of elements contained in *notifications_list*. This must be a positive integer. If `CMM_CMC_NOTIFY_ALL` and `CMM_CMC_NOTIFY_NONE` are used with the *action* parameter, this argument is ignored and not tested.

An application can call the `cmm_cmc_filter()` function as many times as needed. The changes to the filter take effect when the call returns successfully. The filter is evaluated in `cmm_notify_dispatch`(3CMM), so you must define it before calling this dispatching function. An application calling `cmm_cmc_register()` after `cmm_cmc_filter()` does not receive unsolicited notifications.

RETURN VALUES

The `cmm_cmc_filter()` function returns one of the following values:

`CMM_EINVAL`	Invalid argument such as *notification_count* is a `NULL` or *action* is not valid.
`CMM_OK`	Operation succeeds.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `nhcmmd`(1M), `cmm_cmc_register`(3CMM), `cmm_notify_getfd`(3CMM), `cmm_notify_dispatch`(3CMM)

NAME

cmm_cmc_register, cmm_cmc_unregister – register to receive notifications; remove registration and stop receiving notifications

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

typedef struct {
    cmm_cmchanges_t    cmchange;
    cmm_nodeid_t nodeid;
} cmm_cmc_notification_t;

typedef void (*cmm_notify_t)
    (const cmm_cmc_notification_t *change_notification,
     void *client_data);

cmm_error_t cmm_cmc_register(cmm_notify_t const callback, void *
     client_data);

cmm_error_t cmm_cmc_unregister ( );
```

DESCRIPTION

The `cmm_cmc_register()` function enables a system service or application to receive change notifications by registering the callback function indicated.

PARAMETERS

The `cmm_cmc_register()` function takes the following parameters:

callback	A pointer to a callback function defined by the service or application.
client_data	A parameter used by the callback function. Its type and value are defined by the registered service or application. It is passed as an argument to the callback and it is valid in the calling process address space. No sanity check is run on this parameter, as `nhcmmd`(1M) does not know its meaning.

EXTENDED DESCRIPTION

If the `cmm_cmc_register()` function is called while a callback is registered, a `CMM_EEXIST` error is returned because only one function can be registered at a time. To change the registration, `cmm_cmc_unregister()` must be called prior to registering the new node with `cmm_cmc_register()`.

Registration only needs to be done once to receive cluster membership change notifications. When a process attempts to register more than once, the first callback is kept and an error is returned.

The calling process must use the `cmm_notify_getfd`(3CMM) and `cmm_notify_dispatch`(3CMM) functions to receive and dispatch messages from `nhcmmd`(1M). An application defines the list of notifications it receives by calling `cmm_cmc_filter`(3CMM).

Note that the order of callback notifications is the same as that of the Cluster Membership Manager (CMM) notifications; one call to `cmm_notify_dispatch`(3CMM) can lead to several calls to the callback (in fact as many as the number of pending notifications). Within these callbacks, if functions are invoked concerning the state of the cluster (for instance to get the number of nodes in the cluster), the results of the functions do not refer to the state that the cluster is in when the notification has been generated. Instead, the results of the functions refer to the state of the cluster when the function is invoked. In the meantime, some other modifications might have occurred in the cluster.

The callback function is invoked by the same thread as the one that calls the `cmm_notify_dispatch`(3CMM) function. The function is invoked by a library linked to the process. The library communicates with the CMM API that supplies the membership change information passed as an argument to the callback function.

The `cmm_cmc_unregister()` function removes the calling process's registration so that no further delivery of cluster membership change notifications is made. Only the process on which a callback is called can remove the caller's registration. A child or a parent process cannot do this.

If the calling process callback function is active when the unregister request is made, it is not canceled.

In case of `fork()`, the created child process does not inherit the registration from its parent. It has to make its own registration.

RETURN VALUES

The `cmm_cmc_register()` function returns one of the following values:

`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	No `nhcmmd` is currently accessible on the local node.
`CMM_ENOENT`	The maximum number of clients that can register with `cmm_cmc_register` at any one time has been reached.
`CMM_EEXIST`	The calling process has already registered a callback.
`CMM_ENOTSUP`	Unexpected service error. The cluster might be in a critical state.
`CMM_ETIMEDOUT`	The call timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

The `cmm_cmc_unregister()` function returns one of the following values:

CMM_EBUSY	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics. Another possible meaning of CMM_EBUSY is that a call is made to the cmm_cmc_unregister() function when the calling process's callback function is active.
CMM_ECONN	There is no nhcmmd currently accessible on the local node.
CMM_ENOENT	The registration does not exist.
CMM_ENOTSUP	Unexpected service error. The cluster might be in a critical state.
CMM_ETIMEDOUT	The call timeout expired before the action was completed.
CMM_OK	Operation succeeds.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

nhcmmd(1M), fork(2), cmm_cmc_filter(3CMM), cmm_notify_dispatch(3CMM), cmm_notify_getfd(3CMM)

NAME

cmm_cmc_register, cmm_cmc_unregister – register to receive notifications; remove registration and stop receiving notifications

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

typedef struct {
    cmm_cmchanges_t    cmchange;
    cmm_nodeid_t nodeid;
} cmm_cmc_notification_t;

typedef void (*cmm_notify_t)
    (const cmm_cmc_notification_t *change_notification,
     void *client_data);

cmm_error_t cmm_cmc_register(cmm_notify_t const callback, void *
     client_data);

cmm_error_t cmm_cmc_unregister ( );
```

DESCRIPTION

The `cmm_cmc_register()` function enables a system service or application to receive change notifications by registering the callback function indicated.

PARAMETERS

The `cmm_cmc_register()` function takes the following parameters:

callback — A pointer to a callback function defined by the service or application.

client_data — A parameter used by the callback function. Its type and value are defined by the registered service or application. It is passed as an argument to the callback and it is valid in the calling process address space. No sanity check is run on this parameter, as `nhcmmd`(1M) does not know its meaning.

EXTENDED DESCRIPTION

If the `cmm_cmc_register()` function is called while a callback is registered, a `CMM_EEXIST` error is returned because only one function can be registered at a time. To change the registration, `cmm_cmc_unregister()` must be called prior to registering the new node with `cmm_cmc_register()`.

Registration only needs to be done once to receive cluster membership change notifications. When a process attempts to register more than once, the first callback is kept and an error is returned.

The calling process must use the `cmm_notify_getfd`(3CMM) and `cmm_notify_dispatch`(3CMM) functions to receive and dispatch messages from `nhcmmd`(1M). An application defines the list of notifications it receives by calling `cmm_cmc_filter`(3CMM).

Note that the order of callback notifications is the same as that of the Cluster Membership Manager (CMM) notifications; one call to `cmm_notify_dispatch`(3CMM) can lead to several calls to the callback (in fact as many as the number of pending notifications). Within these callbacks, if functions are invoked concerning the state of the cluster (for instance to get the number of nodes in the cluster), the results of the functions do not refer to the state that the cluster is in when the notification has been generated. Instead, the results of the functions refer to the state of the cluster when the function is invoked. In the meantime, some other modifications might have occurred in the cluster.

The callback function is invoked by the same thread as the one that calls the `cmm_notify_dispatch`(3CMM) function. The function is invoked by a library linked to the process. The library communicates with the CMM API that supplies the membership change information passed as an argument to the callback function.

The `cmm_cmc_unregister()` function removes the calling process's registration so that no further delivery of cluster membership change notifications is made. Only the process on which a callback is called can remove the caller's registration. A child or a parent process cannot do this.

If the calling process callback function is active when the unregister request is made, it is not canceled.

In case of `fork()`, the created child process does not inherit the registration from its parent. It has to make its own registration.

RETURN VALUES

The `cmm_cmc_register()` function returns one of the following values:

`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	No `nhcmmd` is currently accessible on the local node.
`CMM_ENOENT`	The maximum number of clients that can register with `cmm_cmc_register` at any one time has been reached.
`CMM_EEXIST`	The calling process has already registered a callback.
`CMM_ENOTSUP`	Unexpected service error. The cluster might be in a critical state.
`CMM_ETIMEDOUT`	The call timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

The `cmm_cmc_unregister()` function returns one of the following values:

CMM_EBUSY	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics. Another possible meaning of CMM_EBUSY is that a call is made to the cmm_cmc_unregister() function when the calling process's callback function is active.
CMM_ECONN	There is no nhcmmd currently accessible on the local node.
CMM_ENOENT	The registration does not exist.
CMM_ENOTSUP	Unexpected service error. The cluster might be in a critical state.
CMM_ETIMEDOUT	The call timeout expired before the action was completed.
CMM_OK	Operation succeeds.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

nhcmmd(1M), fork(2), cmm_cmc_filter(3CMM), cmm_notify_dispatch(3CMM), cmm_notify_getfd(3CMM)

NAME

cmm_config_reload – reload the cluster node table

SYNOPSIS

```
cc [ flag... ] file...  lcgha_cmm  -lrt

#include <cmm.h>

cmm_error_t cmm_config_reload( );
```

DESCRIPTION

The `cmm_config_reload()` function forces the `nhcmmd`(1M) daemon to reload the cluster node table. When a node is added to or removed from this table, the `nhcmmd` daemon must be informed. There are two ways to inform the `nhcmmd` daemon of the changes to the cluster node table: call the `cmm_config_reload()` function or type one of the following:

```
pkill  -HUP nhcmmd
nhcmmstat--creload

kill  -HUP <CMM process Id>
```

This function can only be called from the master node. If it is called from a non-master node it returns a `CMM_EPERM` error. As a result of this call, notifications are sent indicating the modifications occurring in the cluster because of the new configuration read from the cluster node table. See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information on notifications. The supported operations are add a node and remove a node, with the node powered off. The attributes of a node must not be changed.

To remove a node from the cluster, ensure that the node is powered off before changing the cluster nodes table. Only after a node has been powered off should you perform a `cmm_config_reload()`.

RETURN VALUES

The `cmm_config_reload()` function returns one of the following values:

`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	No `nhcmmd` is accessible on the current node.
`CMM_ENOCLUSTER`	Calling node is not yet in a cluster.
`CMM_ENOTSUP`	Unexpected service error.
`CMM_EPERM`	Permission denied. The function was not called from the master node.
`CMM_ETIMEDOUT`	The call timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `nhcmmd`(1M)

NAME

cmm_connect – prepare or test a connection to the Cluster Membership Manager (CMM)

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_connect( const timespec_t timeout);
```

DESCRIPTION

The `cmm_connect()` function is implicit in the first call to the Cluster Membership Manager (CMM) Application Programming Interface (API). You do not need to call this function to create a CMM connection; but use this function to test the availability of a CMM connection, or to set the timeout value. The default timeout is five seconds.

The *timeout* parameter is globally used by the CMM API to signify the maximum amount of time for which a call can block. The type of the timeout is a *timespec_t* and the value must be greater than 0 seconds, 0 nanoseconds. Note that if the value of the timeout is too short, you risk being unable to use the CMM API. This is because every call would fail since the timeout would be expired before the call finished.

Note – This function is not related to cluster information; therefore, can be called from any node - even a node that is not part of a cluster.

RETURN VALUES

The `cmm_connect()` function returns one of the following values:

`CMM_ECONN`	No `nhcmmd`(1M) is currently accessible on the local node.
`CMM_EINVAL`	The given *timeout* is invalid.
`CMM_ENOTSUP`	Unexpected service error occurs.
`CMM_ETIMEDOUT`	Fails to connect before the previous timeout expired.
`CMM_OK`	Operation succeeds.

This call never returns `CMM_ENOCLUSTER`

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `nhcmmd`(1M)

NAME

cmm_disconnect – close a connection between the current calling process and the `nhcmmd` daemon

SYNOPSIS

```
cc [ flag... ] file...  lcgha_cmm  lrt

#include <cmm.h>

cmm_error_t cmm_disconnect( );
```

DESCRIPTION

The `cmm_disconnect()` function closes the connection between the current calling process and the `nhcmmd` daemon. This frees the resources allocated to the client connection. Notifications are no longer managed by the library. If notifications were registered before this function was called, they are no longer sent.

The connection is automatically re-established and resources reallocated when a function that needs the connection is called. However, the configuration of the notifications (callback function, filters, etc) is not recreated. You must reconfigure the notification registration.

Note – If an application or service calls `cmm_disconnect()` when a `cmm_notify_dispatch()` call is being executed, the `cmm_notify_dispatch()` call is terminated.

RETURN VALUES

The `cmm_disconnect()` function returns one of the following values:

`CMM_ENOTSUP`	Unexpected service error or no local `nhcmmd` is accessible.
`CMM_ETIMEDOUT`	The call timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `nhcmmd`(1M), `cmm_connect`(3CMM)

NAME

cmm_master_getinfo, cmm_vicemaster_getinfo – retrieve information about the master node or the vice-master node

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_master_getinfo(cmm_member_t * const member);

cmm_error_t cmm_vicemaster_getinfo(cmm_member_t * const member);
```

DESCRIPTION

The `cmm_master_getinfo()` function returns information about the current master node. The `cmm_vicemaster_getinfo()` function returns information about the current vice-master node. The information returned by these two functions has the same type and meaning as that returned by the `cmm_member_getinfo()` function.

The *member* parameter is a pointer to a member structure where the function stores the member's information, such as its current state.

See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information on the `cmm_member_t` structure.

RETURN VALUES

The `cmm_master_getinfo()` and `cmm_vicemaster_getinfo()` functions return one of the following values:

`CMM_EAGAIN`	The information might be deprecated, because a node has been out of communication with the master node for a period of time.
`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	No `nhcmmd` is currently accessible on the local node.
`CMM_EINVAL`	Invalid parameter. *member* is a `NULL` pointer.
`CMM_ENOCLUSTER`	The calling node is not yet in a cluster.
`CMM_ENOTSUP`	An unexpected service error occurred. The cluster might be in a critical state.
`CMM_ESRCH`	No such member. This return value is only applicable for the vice-master node.
`CMM_ETIMEDOUT`	The call timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

RETURN VALUES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO `Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_mastership_release – trigger a switchover

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_mastership_release( );
```

DESCRIPTION

The `cmm_mastership_release()` function triggers a switchover. This function must be called from the master node. If a service or application attempts to call this function from another node, `CMM_EPERM` is returned.

If the vice-master node is qualified to be master when `cmm_mastership_release()` is called, then this node becomes master. The calling node remains master until the vice-master node has taken the master role.

If no node is qualified to become master when the `cmm_mastership_release()` function is called, `CMM_ECANCELED` is returned.

See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information on the notifications returned in different scenarios.

Any program on the master node can execute this function. No authentication is performed.

RETURN VALUES

The `cmm_mastership_release()` function returns one of the following values:

`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECANCELED`	Operation cancelled. There was no vice-master to take the master role.
`CMM_ECONN`	No `nhcmmd` is currently accessible on the local node.
`CMM_ENOCLUSTER`	Not in cluster.
`CMM_ENOTSUP`	An unexpected service error occurred.
`CMM_EPERM`	Permission denied as the function was not called from a master node.
`CMM_ETIMEDOUT`	The timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_membership_remove`(3CMM), `cmm_member_setqualif`(3CMM)

NAME

cmm_member_getall, cmm_member_getcount – retrieve information on the cluster

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_member_getall (uint32_t const table_size,
     cmm_member_t * const member_table, uint32_t * const member_count);

cmm_error_t cmm_member_getcount (uint32_t * const member_count);
```

DESCRIPTION

The `cmm_member_getall()` function fills *member_table* with information about all nodes in the cluster. There is a table entry for each node. The information in this table is of the same type and meaning as that returned by `cmm_member_getinfo()`. If *member_table* is a null pointer, `cmm_member_getall()` behaves like the `cmm_member_getcount()` function.

The `cmm_member_getcount()` function returns the number of nodes in the cluster, including the node from which the function is called. The value is stored in the area pointed to by *member_count*. See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for further information on the state of the node.

PARAMETERS

The `cmm_member_getall()` function takes the following parameters:

table_size	Specifies the maximum number of entries in *member_table*. The maximum number of entries is 1024.
member_table	A pointer to an array of structures where the requested information is placed.
member_count	Specifies the number of nodes in the cluster.

EXTENDED DESCRIPTION

The process calling the `cmm_member_getall()` and `cmm_member_getcount()` functions allocates and frees all data structures used to return membership information, including the appropriate number of entries in the cluster node table.

If there are more peer nodes than entries in *member table*, the table is not modified, *member_count* is updated, and a `CMM_ERANGE` error is returned. If there are more member entries than peer nodes, the excess member entries are zeroed out.

If requested membership information is temporarily unavailable, as when a switchover is taking place, a `CMM_ENOCLUSTER` error is returned.

The calling process is in charge of allocating the memory and indicating the number of entries by *table_size*.

See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information on the `cmm_member_t` structure.

RETURN VALUES

The `cmm_member_getall()` and `cmm_member_getcount()` functions return one of the following values:

CMM_EAGAIN	The information might be deprecated because the node has been out of communication with the master for a period of time.
CMM_EBUSY	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
CMM_ECONN	No nhcmmd is currently accessible on the local node.
CMM_EINVAL	Invalid argument such as *member_count* is a NULL pointer, or when *table_size* is greated than 1024.
CMM_ENOCLUSTER	The calling node is not yet in a cluster.
CMM_ENOTSUP	An unexpected service error occurred. The cluster might be in a critical state.
CMM_ERANGE	Not enough entries in the member table to provide the requested information.
CMM_ETIMEDOUT	The call timeout expired before the action was completed.
CMM_OK	Operation succeeds.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

Intro(3CMM), cmm_member_getinfo(3CMM)

NAME

cmm_member_getall, cmm_member_getcount – retrieve information on the cluster

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_member_getall (uint32_t const table_size,
     cmm_member_t * const member_table, uint32_t * const member_count);

cmm_error_t cmm_member_getcount (uint32_t * const member_count);
```

DESCRIPTION

The `cmm_member_getall()` function fills *member_table* with information about all nodes in the cluster. There is a table entry for each node. The information in this table is of the same type and meaning as that returned by `cmm_member_getinfo()`. If *member_table* is a null pointer, `cmm_member_getall()` behaves like the `cmm_member_getcount()` function.

The `cmm_member_getcount()` function returns the number of nodes in the cluster, including the node from which the function is called. The value is stored in the area pointed to by *member_count*. See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for further information on the state of the node.

PARAMETERS

The `cmm_member_getall()` function takes the following parameters:

table_size	Specifies the maximum number of entries in *member_table*. The maximum number of entries is 1024.
member_table	A pointer to an array of structures where the requested information is placed.
member_count	Specifies the number of nodes in the cluster.

EXTENDED DESCRIPTION

The process calling the `cmm_member_getall()` and `cmm_member_getcount()` functions allocates and frees all data structures used to return membership information, including the appropriate number of entries in the cluster node table.

If there are more peer nodes than entries in *member table*, the table is not modified, *member_count* is updated, and a `CMM_ERANGE` error is returned. If there are more member entries than peer nodes, the excess member entries are zeroed out.

If requested membership information is temporarily unavailable, as when a switchover is taking place, a `CMM_ENOCLUSTER` error is returned.

The calling process is in charge of allocating the memory and indicating the number of entries by *table_size*.

See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information on the `cmm_member_t` structure.

RETURN VALUES

The `cmm_member_getall()` and `cmm_member_getcount()` functions return one of the following values:

CMM_EAGAIN	The information might be deprecated because the node has been out of communication with the master for a period of time.
CMM_EBUSY	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
CMM_ECONN	No nhcmmd is currently accessible on the local node.
CMM_EINVAL	Invalid argument such as *member_count* is a NULL pointer, or when *table_size* is greated than 1024.
CMM_ENOCLUSTER	The calling node is not yet in a cluster.
CMM_ENOTSUP	An unexpected service error occurred. The cluster might be in a critical state.
CMM_ERANGE	Not enough entries in the member table to provide the requested information.
CMM_ETIMEDOUT	The call timeout expired before the action was completed.
CMM_OK	Operation succeeds.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

Intro(3CMM), cmm_member_getinfo(3CMM)

NAME

cmm_potential_getinfo, cmm_member_getinfo – retrieve information about a peer node

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_potential_getinfo (cmm_nodeid_t const nodeid,
     cmm_member_t * const member);

cmm_error_t cmm_member_getinfo (cmm_nodeid_t const nodeid,
     cmm_member_t * const member);
```

DESCRIPTION

The `cmm_potential_getinfo()` function retrieves the information contained in the `cmm_member_t` structure for a node identified by its *nodeid*. You can use `cmm_potential_getinfo()` to get into any peer node, even if it has the `CMM_OUT_OF_CLUSTER` state.

See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information about the `cmm_member_t` structure.

The `cmm_member_getinfo()` function retrieves the information in the `cmm_member_t` structure for a peer node.

If the requested membership information is temporarily unavailable the operation is retried until it succeeds or a timeout occurs.

In the case of a timeout, the `CMM_ETIMEDOUT` error is returned. If the *nodeid* specified is not in the cluster node table, a `CMM_ESRCH` error is returned.

PARAMETERS

The `cmm_potential_getinfo()` and `cmm_member_getinfo()` functions take the following parameters:

member	Points to the `cmm_member_t` structure, which contains information about the node.
nodeid	Identifies the node on which information is requested.

RETURN VALUES

The `cmm_potential_getinfo()` and `cmm_member_getinfo()` functions return one of the following values:

`CMM_EAGAIN`	The information might no longer be valid, as the node has been out of communication with the master node for a period of time.
`CMM_EBUSY`	The CMM API server is temporarily unable to respond to the requested operation. Wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	The `nhcmmd` daemon cannot be accessed on the current node.

CMM_EINVAL	Invalid argument such as the *member* is a NULL pointer or invalid nodeid.
CMM_ENOCLUSTER	The calling node is not part of any cluster.
CMM_ENOTSUP	Unexpected service error. The cluster might be in a critical state.
CMM_ESRCH	Returned by the cmm_member_getinfo() function if the node is in the local cluster node table but has the CMM_OUT_OF_CLUSTER role. Returned by both cmm_potential_getinfo() and cmm_member_getinfo() functions if the node is not in the local cluster nodes table.
CMM_ETIMEDOUT	The call timeout expired before the action was completed.
CMM_OK	Operation succeeds.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

Intro(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_isdesynchronized, cmm_member_isdisqualified, cmm_member_iseligible, cmm_member_isexcluded, cmm_member_isfrozen, cmm_member_ismaster, cmm_member_isoutofcluster, cmm_member_isqualified, cmm_member_isvicemaster – interpret the status of a member

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

int cmm_member_isdesynchronized(cmm_member_t const * member);

int cmm_member_isdisqualified(cmm_member_t const * member);

int cmm_member_iseligible(cmm_member_t const * member);

int cmm_member_isexcluded(cmm_member_t const * member);

int cmm_member_isfrozen(cmm_member_t const * member);

int cmm_member_ismaster(cmm_member_t const * member);

int cmm_member_isoutofcluster(cmm_member_t const * member);

int cmm_member_isqualified(cmm_member_t const * member);

int cmm_member_isvicemaster(cmm_member_t const * member);
```

DESCRIPTION

These functions enable an application to obtain the status of a peer node. The status information provided includes the membership attributes and role of the cluster as defined in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

PARAMETERS

The `cmm_member_is*()` function takes the following parameter:

member — A pointer to a member structure that contains the member's information, such as a structure filled by `cmm_member_getinfo`(3CMM).

EXTENDED DESCRIPTION

The information provided by the `cmm_member_is*()` functions is as follows:

`cmm_member_isdesynchronized()`
The node is desynchronized if !=0 is returned. The desynchronization flag is set for this node. For the master, !=0 means that it owns the only up-to-date disk; the disks on all other nodes are stale. Remember to check the eligibility of this node to determine if it is a potential master.

`cmm_member_isdisqualified()`
The node is disqualified if !=0 is returned. A node can be disqualified by sending a call to `cmm_member_setqualif()` to set the flag `CMM_MEMBER_DISQUALIFIED`. Check the eligibility of the node to verify that it can become master.

`cmm_member_iseligible()`
The node is a master-eligible node if !=0 is returned.

`cmm_member_isexcluded()`
The node is excluded if !=0 is returned.

`cmm_member_isfrozen()`
The node is frozen if !=0 is returned.

`cmm_member_ismaster()`
The node is the master node if !=0 is returned.

`cmm_member_isoutofcluster()`
The node is not currently participating in cluster services.

`cmm_member_isqualified()`
The node is qualified if !=0 is returned.

`cmm_member_isvicemaster()`
The node is the vice-master if !=0 is returned.

Note – !=0 is returned if, and only if, the node is neither disqualified nor desynchronized. A diskless node can also have this state. Check the eligibility of the node.

RETURN VALUES

The `cmm_member_is*()` functions return one of the following values:

`TRUE` (!=0)	Condition is verified.
`FALSE` (==0)	Condition is not satisfied.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_getinfo`(3CMM)

NAME

cmm_member_seizequalif – requalify current master-eligible node

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_member_seizequalif();
```

DESCRIPTION

The `cmm_member_seizequalif()` function qualifies the current node as the master node when there is no master node.

The `cmm_member_seizequalif()` function must be called from a node that is master-eligible and has the attribute `CMM_ELIGIBLE_MEMBER`. If there is no master in the cluster, `cmm_member_setqualif()` cannot be called. If a node already exists with the attribute `CMM_QUALIFIED_MEMBER`, this call returns `CMM_EPERM`.

EXTENDED DESCRIPTION

There are two outcomes of calling `cmm_member_seizequalif()` from a master-eligible node: either the node becomes master or it reverts to the qualification level it had prior to the `cmm_member_seizequalif()` function. This function returns a `CMM_EPERM` error if a master is already up and running or if the current node is not master-eligible. Note that if the node was previously `CMM_SYNCHRO_NEEDED` (flag S) it will not be elected as master if its former role was master.

See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for further information on qualification levels.

Necessary notifications are sent according to the impact of this call. See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information on possible notifications.

RETURN VALUES

The `cmm_member_seizequalif()` function returns one of the following values:

`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	No `nhcmmd` is currently accessible to the local node.
`CMM_ENOTSUP`	An unexpected service error occurred.
`CMM_EPERM`	Permission denied. Either the function was not called from a master-eligible node, or a master is already running.
`CMM_ETIMEDOUT`	The call timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM), `cmm_member_setqualif`(3CMM)

NAME

cmm_member_setqualif – give a new level of qualification to a node

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_member_setqualif(cmm_nodeid_t const nodeid,
     cmm_qualif_t const new_qualif);
```

DESCRIPTION

The `cmm_member_setqualif()` function assigns a new qualification level to a node. A qualification level is only meaningful for a node with the administrative attribute `CMM_ELIGIBLE_MEMBER` in the `sflag` field of the `cmm_member_t` structure. See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for further information on the `cmm_member_t` structure.

PARAMETERS

The `cmm_member_setqualif()` function takes the following parameters:

nodeid	Specifies the ID of the node whose qualification level is to be changed.
new_qualif	Specifies the new level of qualification assigned to the node.

EXTENDED DESCRIPTION

The `cmm_member_setqualif()` function can only be called from the master node. If an attempt is made to call this function from a node other than the master, a `CMM_EPERM` error is returned.

The *nodeid* given to the `cmm_member_setqualif()` function must be that of a node with the `CMM_ELIGIBLE_MEMBER` attribute. Changing the qualification level of a node in the cluster that does not have this attribute generates a `CMM_EINVAL` returned status. Changing the qualification level of a node with the `CMM_OUT_OF_CLUSTER` state will be successful as it is impossible to know whether or not the node is eligible. If the `nodeid` given to this function as a parameter is that of the master node, a failover occurs and a `MASTER_DEMOTED` notification is sent. `CMM_ENOCLUSTER` is returned until a new master is elected.

The *new_qualif* parameter given to the `cmm_member_setqualif()` function is one of the following parameters:

`CMM_QUALIFIED_MEMBER`	The current node can take any role.
`CMM_DISQUALIFIED_MEMBER`	The current node cannot participate in a master or a vice-master election.
`CMM_SYNCHRO_READY`	The node's disks are synchronized; the vice-master node can become master if necessary. Only meaningful for eligible nodes. A user application should not set this

	flags. CRFS is in charge of setting them and changing them may disrupt the cluster.
CMM_SYNCHRO_NEEDED	The node's disks are not synchronized; the vice-master node can not become master if necessary. Only meaningful for eligible nodes. A user application should not set this flags. CRFS is in charge of setting them and changing them may disrupt the cluster.

The cmm_member_setqualif call is asynchronous; the call's action is not fully completed when the call returns. If a second cmm_member_setqualif call is made to disqualify the master while the first call is still executing but before the master is demoted, both calls return CMM_OK, even though the master is in the process of being demoted. The result is not affected by the second call.

After a cmm_member_setqualif call, for a short period of time the cluster has no master, until the vice-master becomes the new master. Any call to the Cluster Membership Manger (CMM) Application Programming Interface (API) function in this short period of time will return CMM_ENOCLUSTER.

Necessary notifications are sent according to the impact of this call, as described in the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide*.

The value of sflag on the indicated node is immediately updated when a change in the cluster state occurs, so a call to cmm_member_getinfo(3CMM) will reflect the change.

RETURN VALUES

The cmm_member_setqualif() function returns one of the following errors:

CMM_EBUSY	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
CMM_ECONN	No nhcmmd is currently accessible in the local node.
CMM_EINVAL	Invalid parameter. Either the *nodeid* is not that of a master-eligible node or new_qualif is incorrect.
CMM_ENOCLUSTER	The calling node is not yet in a cluster.
CMM_ENOTSUP	An unexpected service error occurred.
CMM_EPERM	Permission denied. The function was not called from the master node.

CMM_ETIMEDOUT	The call timeout expired before the action was completed.
CMM_OK	Operation succeeds.

ATTRIBUTES

Various attributes must be considered.

See attributes(5) for descriptions of the attributes described in the following table:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

Intro(3CMM), cmm_member_getinfo(3CMM), cmm_mastership_release(3CMM), cmm_membership_remove(3CMM)

NAME

cmm_membership_remove – remove peer node

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_membership_remove( );
```

DESCRIPTION

The `cmm_membership_remove()` function removes the node from which this function is called from the cluster. The node is still configured to be in the cluster, but its role has changed. The CMM API is always accessible for an `CMM_OUT_OF_CLUSTER` node.

When the `nhcmmd` daemon detects that a node is no longer part of the cluster, it informs other applications or services that are registered to receive notification. See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information on the notifications returned in different scenarios.

To reintegrate a node into the cluster after the `cmm_membership_remove()` function has been called on the node, restart the node's `nhcmmd` daemon by rebooting the node.

Note that after the `cmm_membership_remove()` call, a node is still to be configured to be in the cluster, but it has the `CMM_OUT_OF_CLUSTER` role. Because it is still configured to be in the cluster, it can access cluster information:

- Functions that retrieve information on the cluster state can still be called by the node
- Functions that modify the cluster state can no longer be called by the node

This is different from a node not being configured for *any* cluster. If the node is not configured for *any* cluster, the `CMM_ENOCLUSTER` value is returned.

Any program running on the node can call `cmm_membership_remove()`. There is no authentication carried out of the program making the call. Calling this function from the master leads to a failover (or `CMM_ECANCELED` if no vice-master can take the mastership role).

RETURN VALUES

The `cmm_membership_remove()` function returns one of the following values:

`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECANCELED`	Operation canceled. The function was called on the master node but the vice-master node was not qualified to become master.
`CMM_ECONN`	No `nhcmmd` is accessible to the current node.

CMM_ENOCLUSTER	The calling node is not yet in a cluster.
CMM_ENOTSUP	An unexpected service error occurred. The cluster might be in a critical state.
CMM_ETIMEDOUT	The call timeout expired before the action was completed.
CMM_OK	Operation succeeds.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

Intro(3CMM), nhcmmd(1M)

NAME

cmm_node_getid – retrieve ID of a node

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_node_getid (cmm_nodeid_t * const me);
```

DESCRIPTION

The `cmm_node_getid()` function retrieves a *nodeid*. This value is accessible even if no master is currently elected. Using the *nodeid*, the node can retrieve information on its status in the cluster.

The *me* parameter is a pointer to a `cmm_nodeid_t` structure. If an error occurs, *me* contains `CMM_INVALID_NODE_ID` and the returned value shows the cause of the error.

Note – This function is not related to cluster information and so it can be called from any peer node, even a node with the `CMM_OUT_OF_CLUSTER` state.

RETURN VALUES

The `cmm_node_getid()` function returns one of the following values:

`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	No `nhcmmd`(1M) is accessible on the current node.
`CMM_EINVAL`	Invalid parameter. *me* is a `NULL` pointer.
`CMM_ENOTSUP`	An unexpected service error occurred. The cluster might be in a critical state.
`CMM_ETIMEDOUT`	The call timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

This call never returns `CMM_ENOCLUSTER`.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO `Intro`(3CMM), `nhcmmd`(1M)

NAME

cmm_notify_dispatch – dispatch cluster membership change messages

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_notify_dispatch( );
```

DESCRIPTION

The `cmm_notify_dispatch()` function processes a cluster membership change control message and invokes the appropriate callback function.

The process uses the `select` or `poll` commands to detect messages arriving from the `nhcmmd` daemon, with at least the file descriptor returned from the `cmm_notify_getfd()` function. When the file descriptor indicates that data must be read, the `cmm_notify__dispatch()` function is called, and the registered callback function is invoked from the same thread that calls the `cmm_notify_dispatch()` function.

If an error occurs on this file descriptor within `poll()` or if the file descriptor is no longer valid, `CMM_EBADF` is returned by `cmm_notify_dispatch()`. Then `cmm_cmc_unregister`(3CMM) must be called and the whole registration must be performed - including calling the `cmm_cmc_register`(3CMM), `cmm_cmc_filter`(3CMM) and `cmm_notify_getfd()` functions.

Note that one call to `cmm_notify_dispatch()` can lead to as many calls to the callback as there are pending notifications. If within this callback, some functions are invoked concerning the state of the cluster (for instance, to get the number of nodes), the result of the function refers to the state of the cluster when the function was invoked. It does not refer to the state of the cluster when the notification was generated. In the meantime, some other modifications could have been applied to the cluster.

RETURN VALUES

The `cmm_notify_dispatch()` function returns one of the following values:

`CMM_EBADF`	Bad file descriptor.
`CMM_ENOENT`	No callback is currently registered.
`CMM_ENOTSUP`	Unexpected service error. Cluster might be in a critical state.
`CMM_OK`	Operation succeeds.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO `Intro`(3CMM), `select`(3C), `poll`(2), `cmm_cmc_filter`(3CMM), `cmm_cmc_register`(3CMM), `cmm_cmc_unregister`(3CMM), `cmm_notify_getfd`(3CMM)

NAME

cmm_notify_getfd – receive cluster membership change messages

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_notify_getfd(int * const fd);
```

DESCRIPTION

The `cmm_notify_getfd()` function stores a file descriptor, in *fd*. This descriptor detects the cluster membership change control messages sent by the `nhcmmd` daemon. System services or applications that require cluster membership change notification delivery must use the `cmm_notify_getfd()` and the `cmm_notify__dispatch()` functions to receive and process messages from the `nhcmmd` daemon.

The *fd* parameter points to the location where the function stores the file descriptor used to receive messages from `nhcmmd`.

EXTENDED DESCRIPTION

Use the `select()` or `poll()` functions and the file descriptor returned from `cmm_notify_getfd()` to detect messages arriving from `nhcmmd`. When the file descriptor indicates that data must be read, `cmm_notify_dispatch()` must be called. This triggers the associated callback registered through `cmm_cmc_register ()`.

If an error occurs on this file descriptor within `poll()` or if the file descriptor is no longer valid, `cmm_notify_dispatch()` returns a `CMM_EBADF` error. The thread that received this error must call the `cmm_cmc_unregister()` and the whole registration process must be performed, that is calling the `cmm_cmc_register()`, `cmm_cmc_filter()`, and `cmm_notify_getfd()` functions.

If the `fork` command is called, the returned file descriptor is automatically closed in the created child process. The file descriptor is only valid within the parent.

If an application calls the `cmm_notify_getfd()` function, it must use the returned file descriptor. The Cluster Membership Manager (CMM) library itself does not monitor events, so if the calling process does not call the `cmm_notify_dispatch()` function when an event occurs, the events accumulate without being handled.

RETURN VALUES

The `cmm_notify_getfd()` function returns one of the following values:

`CMM_EINVAL`	Invalid argument.
`CMM_ENOENT`	No callback is currently registered.
`CMM_OK`	Operation succeeds

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO `Intro`(3CMM), `select`(3C), `nhcmmd`(1M), `fork`(2), `poll`(2), `cmm_cmc_filter`(3CMM), `cmm_cmc_register`(3CMM), `cmm_cmc_unregister`(3CMM), `cmm_notify_dispatch`(3CMM), `select`(3C), `poll`(2)

NAME

cmm_potential_getinfo, cmm_member_getinfo – retrieve information about a peer node

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_potential_getinfo (cmm_nodeid_t const nodeid,
    cmm_member_t * const member);

cmm_error_t cmm_member_getinfo (cmm_nodeid_t const nodeid,
    cmm_member_t * const member);
```

DESCRIPTION

The `cmm_potential_getinfo()` function retrieves the information contained in the `cmm_member_t` structure for a node identified by its *nodeid*. You can use `cmm_potential_getinfo()` to get into any peer node, even if it has the `CMM_OUT_OF_CLUSTER` state.

See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information about the `cmm_member_t` structure.

The `cmm_member_getinfo()` function retrieves the information in the `cmm_member_t` structure for a peer node.

If the requested membership information is temporarily unavailable the operation is retried until it succeeds or a timeout occurs.

In the case of a timeout, the `CMM_ETIMEDOUT` error is returned. If the *nodeid* specified is not in the cluster node table, a `CMM_ESRCH` error is returned.

PARAMETERS

The `cmm_potential_getinfo()` and `cmm_member_getinfo()` functions take the following parameters:

member	Points to the `cmm_member_t` structure, which contains information about the node.
nodeid	Identifies the node on which information is requested.

RETURN VALUES

The `cmm_potential_getinfo()` and `cmm_member_getinfo()` functions return one of the following values:

`CMM_EAGAIN`	The information might no longer be valid, as the node has been out of communication with the master node for a period of time.
`CMM_EBUSY`	The CMM API server is temporarily unable to respond to the requested operation. Wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	The `nhcmmd` daemon cannot be accessed on the current node.

CMM_EINVAL	Invalid argument such as the *member* is a NULL pointer or invalid nodeid.
CMM_ENOCLUSTER	The calling node is not part of any cluster.
CMM_ENOTSUP	Unexpected service error. The cluster might be in a critical state.
CMM_ESRCH	Returned by the cmm_member_getinfo() function if the node is in the local cluster node table but has the CMM_OUT_OF_CLUSTER role. Returned by both cmm_potential_getinfo() and cmm_member_getinfo() functions if the node is not in the local cluster nodes table.
CMM_ETIMEDOUT	The call timeout expired before the action was completed.
CMM_OK	Operation succeeds.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

Intro(3CMM)

NAME

cmm_strerror – get error message string

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

char *cmm_strerror(cmm_error_t errnum);
```

DESCRIPTION

The `cmm_strerror()` function maps the error number in `errnum` to an error message string, and returns a pointer to that string. The returned string must not be overwritten or freed by the calling process.

`errnum` is a string representing an error code.

RETURN VALUES

The `cmm_strerror()` function returns one of the following values:

"No Error"	`errnum` is equal to `CMM_OK`.
"Unknown Error"	`errnum` does not correspond to any error code.
"A string equal to the error code"	`errnum` is equal to the error code.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO

`Intro`(3CMM)

NAME

cmm_master_getinfo, cmm_vicemaster_getinfo – retrieve information about the master node or the vice-master node

SYNOPSIS

```
cc [ flag... ] file...  -lcgha_cmm   -lrt

#include <cmm.h>

cmm_error_t cmm_master_getinfo(cmm_member_t * const member);

cmm_error_t cmm_vicemaster_getinfo(cmm_member_t * const member);
```

DESCRIPTION

The `cmm_master_getinfo()` function returns information about the current master node. The `cmm_vicemaster_getinfo()` function returns information about the current vice-master node. The information returned by these two functions has the same type and meaning as that returned by the `cmm_member_getinfo()` function.

The *member* parameter is a pointer to a member structure where the function stores the member's information, such as its current state.

See the *Netra High Availability Suite Foundation Services 2.1 6/03 CMM Programming Guide* for information on the `cmm_member_t` structure.

RETURN VALUES

The `cmm_master_getinfo()` and `cmm_vicemaster_getinfo()` functions return one of the following values:

`CMM_EAGAIN`	The information might be deprecated, because a node has been out of communication with the master node for a period of time.
`CMM_EBUSY`	The CMM API server is temporarily out of resources to respond to the requested operation. The recommended action is to wait a short time and retry the operation. The length of the waiting must be decided by the user, depending on the application's characteristics.
`CMM_ECONN`	No `nhcmmd` is currently accessible on the local node.
`CMM_EINVAL`	Invalid parameter. *member* is a `NULL` pointer.
`CMM_ENOCLUSTER`	The calling node is not yet in a cluster.
`CMM_ENOTSUP`	An unexpected service error occurred. The cluster might be in a critical state.
`CMM_ESRCH`	No such member. This return value is only applicable for the vice-master node.
`CMM_ETIMEDOUT`	The call timeout expired before the action was completed.
`CMM_OK`	Operation succeeds.

RETURN VALUES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcmd
Interface Stability	Evolving
MT-Level	MT-Safe
Cancel-Safety	Deferred-Cancel-Safe Asynchronous-Cancel-Unsafe

SEE ALSO `Intro`(3CMM), `cmm_member_getinfo`(3CMM)

File Formats

NAME

addon.conf – `nhinstall` configuration file to install additional patches and packages

SYNOPSIS

`addon.conf`

DESCRIPTION

You can configure the `nhinstall` tool to install additional patches and packages by modifying the `addon.conf` file.

It is not mandatory to configure this file. If this file is not configured or not present in the directory containing the configuration files, the `nhinstall` tool assumes there are no additional patches or packages to be installed. This file can be used to upgrade the Foundation Services at a later stage.

The templates for the configuration files are contained in the `/opt/SUNWcgha/config.standard` directory with `.template` extensions. Templates for the `addon.conf` file are specific to the hardware platform type. Copy the necessary `addon.conf` template files to a local directory on the installation server as follows:

```
# mkdir config-file-directory
# export NHOME=/opt/SUNWcgha/config.standard
# cp $NHOME/addon.conf.*.template config-file-directory
```

Note – All the configuration files must be in the same local directory on the installation server.

The `addon.conf` file format is ASCII. Comment lines begin with the comment mark (#). Parameters consist of a keyword followed by an equals (=) sign followed by the parameter value, of the form:

Keyword=Value

Within the *Value*, you can use a slash at the end of a line to indicate that the *Value* is continued on the following line. You can also add comments within a *Value*. For example:

```
PATCH=123456-01 \ #This is the patch number
NHAS -  S USR_SPECIFIC  Y Y \ #This is specific to Foundation Services
```

Each additional patch or package to be installed must be specified in `addon.conf` by using the following parameters:

`PATCH=`*reference dir sub_dir phase scope men diskless* `[ -` *method* `]`

`PACKAGE=`*reference dir sub_dir phase scope men diskless*

reference The patch number or the package name.

dir The directory exported from the installation server and mounted on remote nodes.

To install additional patches or packages from the Solaris distribution on the installation server after the Solaris operating system is installed on the nodes, specify the *dir* as `SOLARIS` and the *phase* as `S`. The additional patches or packages must be in the directory specified for `SOLARIS_DIR` in `env_installation.conf`.

To install additional Foundation Services patches or packages after the installation of the Foundation Services, specify the *dir* as `NHAS` and the *phase* as `F`. The additional patch or package must be in the directory specified for `NHAS2_PRODUCT_DIR` in `env_installation.conf`.

sub_dir The *sub_dir* directory is a subdirectory of *dir* containing the patches or packages. If there is no subdirectory and the package or patch is located in the exported directory, specify "-" for the *sub_dir* parameter.

If you define `NHAS` as the value for the *dir* parameter, the "-" takes one of the following values:

- `NetraHAS2.1/Packages` for a package
- `NetraHAS2.1/Patches` for a patch

If you define `SOLARIS` as the value for the *dir* parameter, the "-" takes one of the following values:

- `Solaris_`*x*`/Product` for a package
- `Solaris_`*x*`/Patches` for a patch

Where *x* is `8` or `9`, depending on the Solaris version installed.

phase Indicates the phase when the patch or package must be installed. The phases are:

`I`	The patch or package is installed during the installation of the Solaris operating system on the master-eligible nodes, and after the `smosservices add` command has run for diskless nodes.
`S`	The patch or package is installed after the Solaris operating system is installed on the master-eligible nodes, and after the `smdiskless add` command has run for diskless nodes.
`F`	The patch or package is installed after the Foundation Services are installed for both the master-eligible nodes and the diskless nodes.

scope Indicates where the package or patch will be installed.

`LOCAL`	Install the package or patch on the root partition of master-eligible node or the diskless node. For diskless nodes, the root partition is `/export/root/`*diskless-node-name* on the master node.
`USR_SPECIFIC`	Install the package or patch in the node's `/usr` directory. For diskless nodes, the `/usr` directory is the `/export/Solaris_x/usr_sparc_all` directory on the master-eligible node. The `basedir` is the default.

	`USR_SOLARIS`	Install the package or patch in the node's `/usr` directory. For diskless nodes, the `/usr` directory is the `/export/Solaris_x` directory on the master-eligible node. The `basedir` is `/usr_sparc.all`.
	`CLONE_OPT`	Install the package or patch in the clone area for diskless nodes. The `/usr` directory is `/export/root/clone/Solaris_x/sun4u`. The `basedir` is `/opt`.
	`SHARED`	Install the package or patch in the shared package directory, that is, `/SUNWcgha/local/export/services` If you specify `SHARED`, the package or patch cannot be installed after the Solaris installation on master-eligible nodes because the shared directory does not exist yet.

Note – The `USR_SPECIFIC`, `USR_SOLARIS`, and `CLONE_OPT` parameters are replaced by `LOCAL` if you are installing the software for a master-eligible node.

men Indicates if a patch or package is to be installed on the master-eligible nodes.

Options are `Y` or `N`.

This parameter is ignored if *scope* is set to `SHARED`.

diskless Indicates if a patch or package is to be installed for a diskless node.

Options are `Y` or `N`.

This parameter is ignored if *scope* is set to `SHARED`.

method Indicates the method used for patch installation. The *method* parameter is optional. If this parameter is not present, the default method of adding patches is either `patchadd` or `smosservice patch`. If the *method* parameter is present, the default method used depends on the value of the *PATCH_WITH_SMOSSERVICE* parameter. The possible methods of patch installation are as follows:

`DEFAULT`	Install patches using the default method
`STANDARD`	Install patches using `patchadd`
`SMOSSERVICE`	Install patches using `smosservice`. Note that you cannot use this method on Solaris 9.
`PATCH_WITH_PKGADD`	Install patch using `pkgadd`. This method is reserved for special patches.

EXAMPLES

This section provides examples of how to use the `addon.conf` file.

EXAMPLE 1 Sample addon.conf File

```
# A patch located on the standard NHAS distribution about
# packages located on /usr (installed after Solaris installation)
PATCH=123456-01   NHAS -  S USR_SPECIFIC  Y Y

# A patch about shared packages located on a user's directory
# and installed after the Foundation Services installation
PATCH=789012-03 /mydir nhas2/mypatchdir F SHARED

# A package located on the standard Solaris distribution and
# installed on the root file system only on the diskless nodes.
PACKAGE=SUNWkvm.u  SOLARIS - S LOCAL N Y

# A package located on a user's directory and installed
# only on the master-eligible nodes after the Solaris installation.
PACKAGE=SUNWsiox.u   /export  Solaris/package S LOCAL Y N
```

EXAMPLE 2 A `PATCH` entry that uses the *dir* and *sub_dir* values

```
PATCH=123456-01 /export patches/Nhas F LOCAL Y Y
```

Where:

- The patch 123456-01 is located at `/export/patches/Nhas`.
- The `/export` directory is the directory that will be shared.

The `nhinstall` tool executes the following commands:

- On the installation server:

  ```
  # share -F nfs /export
  ```

- On remote nodes:

  ```
  # mount server_ip:/export /mnt
  # patchadd -r /mnt/patches/Nhas 123456-01
  ```

EXAMPLE 3 Example 3

To export `/export/patches/Nhas`, the entry will be:

```
PATCH=123456-01 /export/patches/Nhas - S LOCAL Y Y
```

Where "`-`" means that the mount point is where the patch directory is located and there is no subdirectory.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhins

SEE ALSO `env_installation.conf`(4), `cluster_definition.conf`(4), `nhinstall`(1M)

NAME

cluster.conf – SMCT configuration file describing the cluster in terms of nodes, node groups, domains, and services

SYNOPSIS

SMCT_CONFIG_DIR/models/cluster.conf

*smct-config-dir***/models/cluster.conf**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `cluster.conf` configuration file describes the target cluster in terms of nodes, node groups, domains, and the services to be run on each node group. This file uses the configuration elements `shelf`, `board`, `disk`, `ip`, and `network`, which are described in `machine.conf` and `network.conf`.

A pre-configured `cluster.conf` template file for each example hardware configuration is available in the `/opt/SUNWcgha/nhsmct/etc/models/` directory.

The `cluster.conf` configuration file contains the following sections:

- Cluster composition

  The `cluster INVOLVE` block contains high level elements that define the cluster:

  - The configuration element `shelf` describes the shelves that contain the cluster nodes hardware.
  - The configuration element `domain` describes the domain associated to the cluster.
  - The configuration element `nodeGroup` contains a definition of each node group. Each node group is defined in terms of master-eligible, dataless, or diskless, and the supported operating environment.
- Cluster domain definition

  The `domain INVOLVE` block defines the networking parameters associated to the cluster:

  - Subnet definition
  - (Optional) Default router definition

  The `domain USE` block defines the access point to the current master node. This consists of the floating address triplet of the master node.
- Cluster node group definitions

  The `nodeGroup INCLUDE` block defines the set of nodes that belong to the node group.

  The `nodeGroup RUN` block defines the Foundation Services run by the nodes in the node group.
- Configuration element definitions

  The configuration elements define the characteristics for each of the cluster nodes.

PARAMETERS

This section describes the parameters in the `cluster.conf` file:

```
ELEMENT cluster name
                  INVOLVE {shelf name}+ {nodeGroup name}+ domain name

ELEMENT domain name [ id domainid ]
                  [ USE {ip name}+ ]
                  [ INVOLVE {router name}* {network name}* ]

{ ELEMENT nodeGroup name type node-group-type os nodes-group-os arch arch-type
                  [ INCLUDE {nodeGroup name}+ ]
                  RUN {service service-name}+
}+

{ ELEMENT node name [id nodeid]
                  USE board name {disk name}*
}+
```

Note – Parameters specified within square brackets ([]) in the above syntax can be defined either in stage 1 or in stage 3 of the SMCT installation process. For more information, see the *Netra High Availability Suite Foundation Services 2.1 6/03 SMCT Installation Guide*.

- *name*

  ASCII string.

- *nodeid*

  The CMM node ID. The *nodeid* must be a decimal representation of the host part of the IP address specified in the `network.conf` file. If you do not specify a value for the *nodeid*, the SMCT calculates the value based on the IP address specified in the `network.conf` file. For more information, see the `network.conf`(4) man page.

- *node-group-type*

  Type of node group. The type can be one of the following:

  - `MASTER_ELIGIBLE`

    A group of master-eligible nodes.

  - `DISKLESS`

    A group of diskless nodes.

  - `DATALESS`

    A group of dataless nodes.

- *node-group-os*

  The operating system used by the node group. This must be configured to `SOLARIS`.

- *arch-type*

  The architecture type of the node group hardware. By default, this value is `SPARC`.

- *domainid*

  The CMM domain ID. Define this parameter in stage three of the configuration process. For information on the range and format of *domainid*, see the `nhfs.conf`(4) man page.

- *service-name*

  Name of the service list that determines which services are run on the node group. The following service lists are available for each type of node group:

  - Master-eligible node group

    To assign services for a master-eligible node group, use the following service list:

    ```
    NHAS_MASTER_ELIGIBLE [NSM] [RBS] [WDT_MASTER_ELIGIBLE]
    ```

    For the master-eligible node groups, you can assign the following services:

    - `NHAS_MASTER_ELIGIBLE` to install the mandatory Foundation Services.
    - (Optional) `NSM` to install the Node State Manager service.
    - (Optional) `RBS` to install the Reliable Boot Service. The `RBS` option can only be assigned to master-eligible node groups that contain diskless nodes. To enable a diskless node to boot, you must assign the `RBS` option.
    - (Optional) `WDT_MASTER_ELIGIBLE` to install the Watchdog Timer. Use the Watchdog Timer only for Netra servers with hardware watchdogs the LOM level. Netra servers with hardware watchdogs at the OPB level do not require this service. These hardware watchdogs are monitored by the server's software.

  - Dataless node group

    - To assign services for a dataless node group in a cluster running the Foundation Services, use the following service list:

      ```
      NHAS_DATALESS    [WDT_DATALESS]
      ```

      - `NHAS_DATALESS` to install the mandatory Foundation Services.
      - (Optional) `WDT_DATALESS` is the Watchdog Timer for the dataless node group. Use the Watchdog Timer only for Netra servers with hardware watchdogs the LOM level. Netra servers with hardware watchdogs at the OPB level do not require this service. These hardware watchdogs are monitored by the server's software.

    - To assign services for a dataless node group that runs only the CGTP standalone service, use the following service list:

      ```
      CGTP_STANDALONE
      PATCH_DATALESS
      ```

  - Diskless node group

    - To assign services for a diskless node group in a cluster running the Foundation Services, use the following service list:

      `NHAS_DISKLESS` *boot-policy* `[WDT_DISKLESS]`

      - `NHAS_DISKLESS` to install the mandatory Foundation Services.
      - *boot-policy* is one of the following:

MAC_ADDR_POLICY—DHCP static boot policy based on the Ethernet address of the diskless nodes.

STATIC_CLIENT_ID_POLICY—DHCP client ID boot policy.

- (Optional) WDT_DISKLESS is the Watchdog Timer for the diskless node group. Use the Watchdog Timer only for Netra servers with hardware watchdogs the LOM level. Netra servers with hardware watchdogs at the OPB level do not require this service. These hardware watchdogs are monitored by the server's software.

- To assign services for a diskless node group that runs only the CGTP standalone service, use the following service list:

```
CGTP_STANDALONE
boot-policy
PATCH_DISKLESS
```

EXAMPLES

The following are examples of components of the cluster.conf file.

EXAMPLE 1 Defining the Cluster Composition

Example of the cluster composition section of a twelve-node cluster.

```
# Cluster composition
#
ELEMENT cluster 12N_cluster
             INVOLVE shelf shelf_1
             shelf shelf_2
             domain cluster_domain
             nodeGroup master_el
             nodeGroup dataless_T1200
             nodeGroup dataless_T1105
```

EXAMPLE 2 Defining the Cluster Domain

Example of the cluster domain section.

```
# Cluster domain definition
#
# id -> CMM domainId
# ip -> master master-nic0 master-nic1 floating addresses
#
ELEMENT domain cluster_domain id 100
             INVOLVE  network phys-A
                  network phys-B
                      network cgtp
                  network external
                  router default-router
          USE     ip master-cgtp
                  ip master-nic0
                  ip master-nic1
```

EXAMPLE 3 Defining the Master-Eligible Node Group

Example node group and node definition for a master-eligible node group in a four-node cluster.

EXAMPLE 3 Defining the Master-Eligible Node Group *(Continued)*

```
# Master-eligible node group and related nodes definitions
ELEMENT nodeGroup master_el type MASTER_ELIGIBLE os SOLARIS arch SPARC
         INCLUDE nodeGroup diskless
                 node peerNode1-4N
                 node peerNode2-4N
         RUN     service NHAS_MASTER_ELIGIBLE
                 service RBS
#
# Master-eligible node definitions
ELEMENT node peerNode1-4N
        USE board T1105@peerNode1
                disk disk1@peerNode1
#
ELEMENT node peerNode2-4N
        USE board T1105@peerNode2
            disk disk1@peerNode2
```

EXAMPLE 4 Defining a Diskless Node Group

Example of a node group and node definition for a diskless node group in a four-node cluster.

```
# diskless group and related nodes definitions
ELEMENT nodeGroup diskless type DISKLESS os SOLARIS arch SPARC
        INCLUDE node peerNode3-4N
                node peerNode4-4N
              RUN     service NHAS_DISKLESS
                service MAC_ADDR_POLICY
#
# diskless nodes definitions
ELEMENT node peerNode3-4N
              USE board T1105@peerNode3
#
ELEMENT node peerNode4-4N
              USE board T1105@peerNode4
```

EXAMPLE 5 Defining a Dataless Node Group

Example of a node group and node definition for a dataless node group in a twelve-node cluster.

```
# Node Groups definitions
#
# Dataless group and related nodes definitions
ELEMENT nodeGroup dataless_T1200 type DATALESS os SOLARIS arch SPARC
        INCLUDE  node peerNode3-12N
                 node peerNode4-12N
                 node peerNode5-12N
                 node peerNode6-12N
        RUN      service NHAS_DATALESS
#
# Dataless nodes definitions
ELEMENT node peerNode3-12N
```

EXAMPLE 5 Defining a Dataless Node Group *(Continued)*

```
        USE  board T1200@peerNode3
             disk disk1@peerNode3
#
ELEMENT node peerNode4-12N
        USE  board T1200@peerNode4
             disk disk1@peerNode4
#
ELEMENT node peerNode5-12N
        USE  board T1200@peerNode5
             disk disk1@peerNode5
#
ELEMENT node peerNode6-12N
        USE  board T1200@peerNode6
             disk disk1@peerNode6
```

EXAMPLE 6 Defining CGTP Standalone in Diskless Node Group

Example of a diskless node group with CGTP standalone.

```
ELEMENT nodeGroup standalone_diskless type DISKLESS os SOLARIS arch SPARC
INCLUDE node peerNode3
INCLUDE node peerNode4
RUN service CGTP_STANDALONE
    service MAC_ADDR_POLICY
    service PATCH_DISKLESS
```

EXAMPLE 7 Defining CGTP Standalone for Dataless Nodes

Example of a dataless node group with CGTP standalone.

```
ELEMENT nodeGroup standalone_dataless type DATALESS os SOLARIS arch SPARC
INCLUDE node peerNode3
INCLUDE node peerNode4
RUN service CGTP_STANDALONE
    service PATCH_DATALESS
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`cluster_nodes_table`(4), `nhfs.conf`(4), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M)

Netra High Availability Suite Foundation Services 2.1 6/03 SMCT Installation Guide

NAME

cluster_definition.conf – `nhinstall` configuration file to define the cluster

SYNOPSIS

`cluster_definition.conf`

DESCRIPTION

Configure the `cluster_definition.conf` file to define the nodes of your cluster for the `nhinstall` tool. The `nhinstall` tool does not support the installation of dataless nodes.

The templates for the configuration files are contained in the `/opt/SUNWcgha/config.standard` directory with `.template` extensions. Copy the configuration files to a local directory on the installation server as follows:

```
# mkdir config-file-directory
# export NHOME=/opt/SUNWcgha/config.standard
# cd config-file-directory
# cp $NHOME/cluster_definition.conf.template cluster_definition.conf
```

Note – All the configuration files must be in the same local directory on the installation server.

The `cluster_definition.conf` file format is ASCII. Comment lines begin with the comment mark (#). Parameters consist of a keyword followed by an equals (=) sign followed by the parameter value, of the form:

Keyword=Value

Within the *Value*, you can use a slash at the end of a line to indicate that the *Value* is continued on the following line. You can also add comments within a *Value*. For example:

```
PUBLIC_NETWORK=255.255.255.0 \ #This is the netmask value
192.168.0.0 \ #This is the subnet value
```

The following *Keyword* and *Value* parameters are supported:

`CLUSTER_ID`
The cluster ID used to assign IP addresses to all the nodes in the cluster. This parameter must be a value between 1 and 254. For example:

```
CLUSTER_ID=250
```

It is mandatory to define this parameter.

`PASSWORD`
The password for the superuser set for all nodes in the cluster. Set this password even if you manually install the Solaris operating system on the master-eligible nodes because this password is required by the `nhinstall` tool to execute the commands that create the diskless environment.

By default, the password is `sunrules`.

`MEN_INTERFACES`
The network interfaces used for Carrier Grade Transfer Protocol (CGTP) on master-eligible nodes. The parameter has the following format:

MEN_INTERFACES=*nic0 nic1*

nic0 is the name of the first network interface.
nic1 is the name of the second network interface.

It is mandatory to define this parameter.

NMEN_INTERFACES

The network interfaces used for CGTP on diskless nodes. The parameter has the following format:

NMEN_INTERFACES=*nic0 nic1*

nic0 is the name of the first network interface.
nic1 is the name of the second network interface.

It is mandatory to define this parameter if diskless nodes are defined.

LOGICAL_SLICE_SUPPORT

Install the volume management feature of the Solaris operating system. Options are YES and NO. The default is NO.

If LOGICAL_SLICE_SUPPORT is set to YES, the volume management feature of the Solaris operating system is configured for managing replicated partitions and their associated bitmap partitions.

The volume management feature must be used for Netra 20 hardware because the disk scanning mechanism does not guarantee that the slot position of the disk provides a unique and reproducible unit number, for example, when some disks are plugged on FC-AL (Fibre Channel-Arbitrated Loop).

If you are using FC-AL disks on Netra 20 servers, you must do one of the following:

- Set SOLARIS_INSTALL to YES in the env_installation.conf file.

  Set LOGICAL_SLICE_SUPPORT to YES.

  Define a partition with the mount point set to replica by using the SLICE parameter.

  Complete the configuration and run the nhinstall(1M) tool.
- Set SOLARIS_INSTALL to NO in the env_installation.conf file.

  Set LOGICAL_SLICE_SUPPORT to NO.

  Install the Solaris operating system yourself on the master-eligible nodes configured with the support of the Solaris Volume Manager (Solaris 9) or Solstice DiskSuite 4.2.1 (Solaris 8).

  Create a disk partition with the attribute replica. This partition is reserved for storing the metadevice database.

  Complete the configuration and run the nhinstall(1M) tool.

SERIALIZE_SYNC
Determine how disk synchronization is performed. Values are YES and NO. The default value is NO.

Synchronization is necessary following a switchover or the vice-master booting, or when you request a full replication. In these circumstances, if SERIALIZE_SYNC is set to NO, Reliable NFS starts the synchonization of all slices at the same time. If SERIALIZE_SYNC is set to YES, slices are synchronized one slice at a time. This reduces the network and disk overhead but increases the time it takes for the vice-master to synchronize with the master. During this time, the vice-master is not eligible to take on the role of master.

USE_WDT
Install and configure the Watchdog Timer packages. Options are YES and NO. The default is NO.

Set the USE_WDT parameter to YES only if you are using Netra servers with hardware watchdogs at the LOM level. When this parameter is set to YES, the Watchdog Timer is installed and configured. In this case, you must install LOM packages. You can install these packages using the addon.conf file. For more information, see the addon.conf(4) man page.

Set the USE_WDT parameter to NO if you are using Netra servers with hardware watchdogs at the OPB level or if you do not want to install the watchdog timer. OPB-level hardware watchdogs are monitored by the server's software.

EXTERNAL_ACCESS
Creates a hostname and an external IP address for the node that has the master role. This IP address is the same for either master-eligible node when that node takes on the master role. When a node takes on the master role, this IP address and hostname are assigned to the master node.

The EXTERNAL_ACCESS parameter has the following format:

EXTERNAL_ACCESS=*hostname IP-address INTERFACE*

- *hostname* is the external host name given to the master node.
- *IP-address* is the external IP address given with the master node.
- *NIC* is the physical network interface to which the IP address is to be associated.

VENDOR_TYPE
List of platform names of diskless nodes to boot. This parameter can have multiple entries. All entries will be used when the nhinstall tool configures the DHCP.

The VENDOR_TYPE parameter enables you to add a string defining the vendor type sent by a diskless node when issuing a DHCP boot request. This enables you to include boards for diskless nodes that are not part of the tested hardware.

The predefined platform names are:

- SUNW.UltraSPARC-IIi-cEngine
- SUNW.UltraSPARC-IIi-Netract

- SUNW.UltraSPARCengine_CP-60
- SUNW.UltraAX-i2
- SUNW.NetraCT-410
- SUNW.NetraCT-810

To add new vendor type:

```
VENDOR_TYPE=new_vendor_type
```

USE_CGTP
Install the CGTP on the cluster. Options are YES and NO. The default is YES.

If USE_CGTP is set to NO, the CGTP packages and patches are not installed. Only the first network interface is considered in the definition of NODE. The CGTP_POSTFIX, if specified, is ignored. In this case, you configure a single network link and your cluster network is not redundant.

PATCH_WITH_SMOSSERVICE
Install diskless patches with smosservice. Options are YES and NO. The default is NO.

If PATCH_WITH_SMOSSERVICE is set to YES, the smosservice patch command installs the patches listed in the addon.conf file and installed on the diskless environment after the Solaris installation. This installation corresponds to phase I or S as described in the addon.conf file. The patchadd command installs all patches applied during phase F. The patchadd command is also used to install any temporary patches, that is patches with names prefixed T.

Note –

- Do not use PATCH_WITH_SMOSSERVICE on the Solaris 9 operating system.
- You can supersede the patch installation method set with PATCH_WITH_SMOSSERVICE on a per patch basis. For more information, see the addon.conf(4) man page.

IDE_SUPPORT
Configure the Sun StorEdge Network Data Replicator (SNDR) to support IDE disks. Options are YES and NO. The default is NO. For example:

```
IDE_SUPPORT=YES
```

RESTRICT_RHOSTS
Restrict the master-eligible nodes from connecting to each other remotely. Options are YES and NO. The default is NO.

If RESTRICT_RHOSTS is set to YES, only the installation server can connect remotely to the master-eligible nodes with rsh and rcp commands. The master-eligible nodes cannot connect to each other remotely. Therefore, you will not be able to run nhadm(1M) commands such as synccheck and syncgen on the master-eligible nodes.

If RESTRICT_RHOSTS is set to NO, the master-eligible nodes can connect to each other remotely with the rsh and rcp commands. The installation server can also connect to the master-eligible nodes remotely.

DISKLESS_BOOT_POLICY
Define the boot policy of the diskless nodes based on a boot policy. Options are DHCP_STATIC, DHCP_DYNAMIC, and DHCP_CLIENT_ID. The default is DHCP_DYNAMIC.

To choose the DHCP dynamic boot policy, specify DHCP_DYNAMIC. In this case, the IP addresses are attributed randomly to diskless nodes from a pool of IP addresses.

To choose the DHCP static boot policy, specify DHCP_STATIC. In this case, the IP addresses are attributed according to the Ethernet address of the diskless nodes. If DHCP_STATIC is selected, the *MAC0* and *MAC1* attributes for diskless nodes must be set in the NODE parameter. Also, in the NODE parameter, the *NIC0* and *NIC1* values can be set if the default values specified by NMEN_INTERFACE need to be superseded.

To choose the DHCP client ID boot policy, specify DHCP_CLIENT_ID. In this case, the IP addresses are attributed to a diskless node based on its client ID. This address assignment scheme is relevant only for CompactPCI servers.

REPLICATED_DHCP_FILES
Select the location of the DHCP configuration files. Options are YES and NO. The default is YES.

If REPLICATED_DHCP_FILES is set to YES, the DHCP configuration files are located on a replicated partition and are shared by the master-eligible nodes. If REPLICATED_DHCP_FILES is set to NO, the DHCP configuration files are duplicated on a local partition of each master-eligible node.

CLUSTER_NETWORK
Define the class of IP addresses for your cluster network. The cluster network can have IP addresses of any class. The parameter has the following format:

CLUSTER_NETWORK=*netmask nic0-subnet nic1-subnet cgtp-subnet*

- *netmask* is the mask common to all subnets configured in the /etc/netmasks file, for example, 255.255.0.0.
- *nic0-subnet* is the subnet of the first network interface, *NIC0*. This subnet is configured in the /etc/netmasks file, for example, 172.15.0.0.
- *nic1-subnet* is the subnet of the second network interface, *NIC1*. This subnet is configured in the /etc/netmasks file, for example, 172.16.0.0.
- *cgtp-subnet* is the subnet of the virtual network interface, *cgtp0*. This subnet is configured in the /etc/netmasks file, for example, 172.17.0.0.

By default, class C IP addresses are used as follows:

CLUSTER_NETWORK=255.255.255.0 10.*clusterid*.1.0 10.*clusterid*.2.0 10.*clusterid*.3.0

Where *clusterid* is the value of the CLUSTER_ID parameter.

DISKLESS_TYPE
Define whether the diskless nodes are to be installed in a cluster or as standalone nodes running only CGTP. Options are CLUSTER and STANDALONE. The default is CLUSTER.

If you specify the CLUSTER option, the diskless nodes are installed with all the Foundation Services. The nodes are configured to be part of the cluster.

If you specify the STANDALONE option, the diskless nodes are installed with the CGTP only. The nodes are configured to run as standalone nodes that do not run all the Foundation Services. The two master-eligible nodes are installed with the Foundation Services so that the standalone diskless node can be booted by the master node.

NODE

Define each node. It is mandatory to define this parameter.

There is an entry for each node and each entry has the following format:

NODE=*nodeid* {*MAC0*|*client-id*|-} {*MAC1*|-} {*name*|-} {*NIC0*|-} {*NIC1*|-} *public-name public-IP public-NIC*

- *nodeid*

  The ID of the node used to define IP addresses for the node. This option is mandatory.

- *MAC0* | *client-id* | -

  The Ethernet address of the first network interface of the node. This option is mandatory.

  - For master-eligible nodes, this address is required to boot the master-eligible node from the installation server.
  - For a diskless node, whether this value is required depends on the DHCP boot policy used:

    For the DHCP static boot policy, when the DISKLESS_BOOT_POLICY is set to DHCP_STATIC, specify the Ethernet address of the first network interface of the node.

    For the DHCP dynamic boot policy, when the DISKLESS_BOOT_POLICY is set to DHCP_DYNAMIC, the *MAC0* address is ignored and you must specify a hyphen (-).

    For the DHCP client ID boot policy, when the DISKLESS_BOOT_POLICY is set to DHCP_CLIENT_ID, specify the client ID as a string in double quotation marks. You can insert a hexadecimal value, for example "/00999:88:05".

    Note – If you want to include the back slash character, you must include it twice.

- *MAC1*

  The Ethernet address of the second network interface of the node. This option is mandatory.

  - For master-eligible nodes, the *MAC1* address is ignored and you must specify a hyphen (-).

- For a diskless node, this value is required depending on the DHCP boot policy used:

  For the DHCP static boot policy, when the `DISKLESS_BOOT_POLICY` is set to `DHCP_STATIC`, specify the Ethernet address of the second network interface of the node.

  For the DHCP dynamic boot policy, when the `DISKLESS_BOOT_POLICY` is set to `DHCP_DYNAMIC`, the *MAC1* address is ignored and you must specify a hyphen (-).

  For the DHCP client ID boot policy, when the `DISKLESS_BOOT_POLICY` is set to `DHCP_CLIENT_ID`, the *MAC1* address is ignored and you must specify a hyphen (-).

- *name*

  Name of the node. By default, the names are assigned as follows:

  - For a master-eligible node, `MEN-C`*clusterid*`-N`*nodeid*.
  - For a diskless node, `NMEN-C`*clusterid*`-N`*nodeid*.

  Do not use underscores ("_") when naming a node. For more information, see `hosts`(4).

- *NIC0*

  Name of the first network interface. This parameter can be set for a diskless node with the DHCP static boot policy only. If you are using the DHCP static boot policy and you want to use the default value, which is the first value defined by the `NMEN_INTERFACES` parameter, specify a hyphen (-).

  For master-eligible nodes and for diskless nodes with other types of DHCP boot policies, the *NIC0* address is ignored. In these cases, you must specify a hyphen (-).

- *NIC1*

  Name of the second network interface. This parameter can be set for a diskless node with the DHCP static boot policy. If you are using the DHCP static boot policy and you want to use the default value, which is the second value defined by the `NMEN_INTERFACES` parameter, specify a hyphen (-).

  For master-eligible nodes and for diskless nodes with other types of DHCP boot policies, the *NIC0* address is ignored. In these cases, you must specify a hyphen (-).

- *public-name*

  Name of the node on the public network different from the name defined with the *name* parameter. If `PUBLIC_NETWORK` is not defined, the *public-name* is ignored.

- *public-ip*

IP address of the node on the public network. If PUBLIC_NETWORK is not defined, the *public-ip* is ignored.

- *public-nic*

  Network interface for the node supporting the public network. This can be either a physical network interface or an alias. If PUBLIC_NETWORK is not defined, the *public-nic* is ignored.

Example 1:

```
NODE=10 08:00:20:f9:c5:54 - node10
NODE=20 08:00:20:f9:a8:12 - node20
NODE=30 -                           - node30
NODE=40 -                           - node40
```

Example 2:

```
NODE=10 08:00:20:f9:c5:54 - node10 - - FSNode1 192.168.12.5 hme1:5
NODE=20 08:00:20:f9:a8:12 - node20 - - FSNode2 192.168.12.6 hme1:101
NODE=30 -                           - node30
NODE=40 -                           - node40
```

The first two entries must be the IDs for the master-eligible nodes. The nhinstall tool first installs the product on the first master-eligible node defined and then on the second master-eligible node. The remaining entries define the diskless nodes.

To add diskless nodes to a cluster that is already running, add the definitions for the new nodes by using the NODE parameter and run the nhinstall command with the add option. For information on the nhinstall command to add diskless nodes to the cluster, see the nhinstall(1M) man page.

PUBLIC_NETWORK

Define the public network IP addresses and netmasks for the cluster. The parameter has the following format:

PUBLIC_NETWORK=*netmask subnet*

- *netmask* is the mask for the subnet configured in the /etc/netmasks file. For example: 255.255.255.0
- *subnet* is the public subnet configured in the /etc/netmasks file. For example: 192.168.0.0

Note – The PUBLIC_NETWORK parameter also configures the network interface of the installation server. Therefore, the SERVER_IP in the env_installation.conf file is an IP address that is part of the same *subnet* as defined for the PUBLIC_NETWORK.

DEFAULT_ROUTER_IP

Defines the IP address of the default router for the public network. The DEFAULT_ROUTER_IP is set to an IP address.

DEFAULT_ROUTER_IP=*IP address*

The default is the public IP address of the installation server.

`NIC0_POSTFIX`, `NIC1_POSTFIX`, `CGTP_POSTFIX`

Defines a suffix string for each host name. These definitions can be set to change the suffix for each host name related to a specific network interface. By default the suffixes are:

- `NIC0_POSTFIX=""`

  Therefore, by default, the host name for the *NIC0* interface has no suffix.
- `NIC1_POSTFIX="-nic1"`
- `CGTP_POSTFIX="-cgtp"`

If you specify all the suffixes, as in the following example, the `nhinstall` tool creates two host names in the `/etc/hosts` file—one host name with the suffix and one host name without the suffix with respect to the *NIC0* interface.

For example, if the suffixes are:

```
NIC0_POSTFIX="-mynic0"
NIC1_POSTFIX="-mynic1"
CGTP_POSTFIX="-mycgtp"
```

The resulting `/etc/hosts` file will include the following type of entry for each node:

```
10.250.1.10 netraMEN1 netraMEN1-mynic0
10.250.1.20 netraMEN2 netraMEN2-mynic0
10.250.1.30 netraDISKLESS1 netraDISKLESS1-mynic0
```

Note – Do not use underscores ("_") within a suffix. For more information, see `hosts`(4).

`DIRECT_LINK`

You can prevent the occurence of two master nodes in one cluster, by connecting the serial ports of the two master-eligible nodes and defining the `DIRECT_LINK` parameter. By default, this parameter is not configured.

The `DIRECT_LINK` parameter has the following format:

`DIRECT_LINK=`*MEN1-serial-device MEN2-serial-device speed* [*heartbeat-in-seconds*]

- *MEN1-serial-device* is the serial port on the first master-eligible node to use to connect to the second master-eligible node, for example, `/dev/ttya`
- *MEN2-serial-device* is the serial port on the second master-eligible node to use to connect to the second master-eligible node, for example, `/dev/ttya`
- *speed* is the serial line speed. Valid values for *speed* are 38400, 57600, 76800, and 115200.
- *heartbeat-in-seconds* is the frequency of the heartbeat checking the link between the two master-eligible nodes. The default value for heartbeat is 20 seconds.

`BITMAP_IN_MEMORY`

Define where the scoreboard bitmaps of the shared partitions are stored.

If you choose the option YES, the scoreboard bitmaps are configured to be stored in memory. In this case, changes are written to the disk only when the node is shut down. This provides better performance on the nodes. However, if both master-eligible nodes fail, the disks must be resynchronized.

Alternatively, if you choose the option NO, the scoreboard bitmaps are configured to write changes to the disk at each update.

For compatibility with previous releases, the default is NO:

```
BITMAP_IN_MEMORY=NO
```

NFS_USER_DIR_NOAC

Define the NFS noac option for remote mounted directories. The noac option suppresses data and attributes caching.

If you choose YES, the noac option is configured when mounting remote directories. In this case, all data is retrieved from the master node disk. Data and attribute caching is suppressed.

Alternatively, if you choose NO, the noac option is not configured. In this case, data is cached on the local node.

Use the noac option if the impact on performance is acceptable.

For compatibility with previous releases, the default is YES, for example:

```
NFS_USER_DIR_NOAC=YES
```

SLICE_SYNC_TYPE

Define how the replicated partitions are synchronized. Options are:

FS	Only blocks that contain data are replicated. Choose this option for faster synchronization.
RAW	All blocks are replicated. Choose this option for slower synchronization.

The default option for SLICE_SYNC_TYPE is FS.

```
SLICE_SYNC_TYPE=FS
```

CHECK_REPLICATED_SLICES

Define whether the replicated partition sanity check is activated. Options are YES and NO. The default is NO.

If CHECK_REPLICATED_SLICES is set to YES, the sanity check is activated. If CHECK_REPLICATED_SLICES is set to NO, the sanity check is not activated.

MASTER_LOSS_DETECTION

Define if the absense of a master node in the cluster must be detected by diskless nodes. Options are YES and NO. The default is YES.

If MASTER_LOSS_DETECTION is set to YES, the absense of a master node in the cluster is detected by the diskless nodes and if such an absense occurs, the diskless nodes are rebooted. If MASTER_LOSS_DETECTION is NO, the absense of a master node in the cluster is not detected by diskless nodes.

SLICE

Defines the disk partitioning. There is an entry for every partition on the disk, and each entry has the following format:

SLICE=*name size mount-point bitmap option*

- *name*

  The name of the disk partition. For example, c0t0d0s0. The *name* can also be a metadevice name if you have already installed the Solaris operating system configured with the volume management feature. However, slice 2 is reserved.

- *size*

  The size of the partition in Mbytes. For slice 7, the size can be set to free unless the partition is replicated.

- *mount-point*

  The name of the mount point of the partition. If the partition is replicated, the mount point must be set to unnamed.

  If the partition is reserved for metadevice database storage, the mount point must be set to replica. This is because one partition is required to have the replica attribute when LOGICAL_SLICE_SUPPORT is set to YES. See Example 3 in this man page.

- *bitmap*

  The name of the replicated partition. If the partition is not replicated, specify a hyphen (-).

- *option*

  The mount option.

  For further details about *name*, *size*, *mount-point*, and *option*, see the Solaris JumpStart™ documentation.

```
SLICE=c0t0d0s0 2048 /               -        logging
SLICE=c0t0d0s1 1024 swap            -        -
SLICE=c0t0d0s3 2048 /export         c0t0d0s5 logging
SLICE=c0t0d0s4 2048 /SUNWcgha/local c0t0d0s6 logging
SLICE=c0t0d0s5 3    unnamed         -        -
SLICE=c0t0d0s6 3    unnamed         -        -
SLICE=c0t0d0s7 free /test1          -        logging
```

The following disk partitions are mandatory:

- The root partition, /
- The /SUNWcgha/local partition

- If diskless nodes are configured, the `/export` partition

`EXPORTED`
The directory to be created and exported on the master–eligible node. There is an entry for every directory to be exported. Each entry has the following format:

`EXPORTED=`*name*

name is the directory to be exported. However, do not specify the following directories:

- `/SUNWcgha/local/export` because the `nhinstall` tool automatically creates this directory
- `/export` if a diskless environment is required, the `nhinstall` tool automatically creates this directory

Note – Exported directories must be on a replicated partition because these directories must be accessible regardless of the node that is master. Exporting a non-replicated directory results in errors during a switchover.

`MOUNTED`
The mount point on the master–eligible node of the directory that is mounted on the master-eligible nodes and the diskless nodes. There is an entry for each mount point, and each entry has the following format:

`MOUNTED=`*mounting-point remote-dir*

- *mounting-point*

  The mount point on each node.

- *remote-dir*

  The directory name on the master-eligible node.

The following directories are automatically created by the `nhinstall` tool. Do not define them.

- The `/SUNWcgha/remote` directory, which is the mount point for the *men_name*`:/SUNWcgha/local/export/data` directory.
- The `/SUNWcgha/services` directory, which is the mount point for the *men_name*`:/SUNWcgha/local/export/services/ha_v1/opt`.
- The `/SUNWcgha/swdb` directory, which is the mount point for the *men_name*`:/SUNWcgha/local/export/services` directory.

Where *men_name* is the host name of the master-eligible nodes.

Caution – For remote directories, define only child directories that were previously exported, that is, directories on a replicated partition.

Do not define `/export` or its subdirectories because it will conflict with the management of diskless nodes.

DATA_MGT_POLICY

Define how the cluster behaves when the vice-master node starts up while the master node is down. Options are:

INTEGRITY	The vice-master waits for the old master to rejoin the cluster before it takes the master role. This ensures that the cluster uses the most up-to-date data.
AVAILABILITY	The vice-master does not wait for the old master to rejoin the cluster before it takes the master role. Data written to the master while the vice-master is down is lost.
ADAPTABILITY	The vice-master checks the disk synchronization state. If the state is not synchronized, that is the state returned by `nhcmmstat` is `synchro:NEEDED`, the vice-master waits for the old master to rejoin the cluster. If the state is synchronized, that is the state returned by `nhcmmstat` is `synchro:READY`, the vice-master is elected as the new master.

The default value for DATA_MGT_POLICY is INTEGRITY.

```
DATA_MGT_POLICY=INTEGRITY
```

SYNC_FLAG

Delays disk synchronization at startup. Options are YES and NO. The default is YES.

If SYNC_FLAG is set to NO, you delay the start of disk synchronization until you use the nhenablesync command. For more information, see the nhenablesync(1M) man page.

MEN_OS_REFERENCE

Defines the profile of the operating system to be used. This parameter is used when the automatic release detection based on the /etc/release file does not enable you to determine the correct operating system version where the Solaris distribution has been updated but there is no change to the Solaris release identification information. Current options are:

S8U7	Solaris 8 2/02 s28s_u7wos_08a SPARC
S8U7_108528-21	Solaris 8 2/02 s28_uwos_08a SPARC — patched version of the Solaris operating system which is reserved for use with Netra CT 820 hardware
S8PSR3	Solaris 8 HW 7/03 s28s_hw3wos_05a SPARC
S9HWPL3	Solaris 9 s9_58shwp13 SPARC
S9U1	Solaris 9 9/02 s9s_u1wos_08b SPARC

There is no default value for MEN_OS_REFERENCE. The option is automatically detected based on the contents of the /etc/release file.

DISKLESS_OS_REFERENCE

Defines the profile of the Solaris operating system used in the diskless node environment. This parameter is used when automatic release detection based on the contents of the /etc/release file does not permit nhinstall to determine the correct version of the operating system. This can occur when the Solaris distribution has been updated but the release identification information has not changed. Options are:

S8U7	Solaris 8 2/02 s28s_u7wos_08a SPARC
S8U7_108528-21	Solaris 8 2/02 s28_uwos_08a SPARC — patched version of the Solaris operating system which is reserved for use with Netra CT 820 hardware
S8PSR3	Solaris 8 HW 7/03 s28s_hw3wos_05a SPARC
S9HWPL3	Solaris 9 s9_58shwp13 SPARC
S9U1	Solaris 9 9/02 s9s_u1wos_08b SPARC

There is no default value for MEN_OS_REFERENCE. The option is automatically detected based on the contents of the /etc/release file.

EXAMPLES

EXAMPLE 1 Example of a standard configuration for a cluster with two master-eligible nodes and one diskless node

```
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
#
#
# The usage of shell variables is authorized.
# This file is sourced by the nhinstall script.
#

#-----------------------------------------------------
#
#  This file enables you to define the cluster
#  environment.
#
#-----------------------------------------------------

CLUSTER_ID=250

ARCH=sun4u

MEN_INTERFACES=hme0 hme1

NMEN_INTERFACES=eri0 eri1

LOGICAL_SLICE_SUPPORT=NO

USE_CGTP=YES

RESTRICT_RHOSTS=NO

NODE=10 08:00:20:f9:b3:6a
```

EXAMPLE 1 Example of a standard configuration for a cluster with two master-eligible nodes and one diskless node *(Continued)*

```
NODE=20 08:00:20:f9:aa:66
NODE=30 -                             - node30

SLICE=c0t0d0s0 2048 /               -        logging
SLICE=c0t0d0s1 1024 swap            -        -
SLICE=c0t0d0s3 2048 /export         c0t0d0s5 logging
SLICE=c0t0d0s4 2048 /SUNWcgha/local c0t0d0s6 logging
SLICE=c0t0d0s5 3    unnamed         -        -
SLICE=c0t0d0s6 3    unnamed         -        -
SLICE=c0t0d0s7 free /test1          -        logging
```

EXAMPLE 2 Example of a configuration for a cluster with no diskless nodes

If you have only two master-eligible nodes in your cluster, the disk layout and disk partitioning are different from those of a cluster with one diskless node or more.

The order of the partitions differs from the order in the preceding example as follows:

- There is no `/export` partition. Instead, there are two user partitions—one replicated and one not replicated.
- There is no bitmap partition associated to the `/export` partition.
- The user directory is defined as exported on both partitions, and the mount points on the user directory are accessible from both nodes.

```
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
#
#
# The usage of shell variables is authorized.
# This file is sourced by the nhinstall script.
#

#-----------------------------------------------------
#
#  This file enables you to define the cluster
#  environment.
#
#-----------------------------------------------------

CLUSTER_ID=250

ARCH=sun4u

MEN_INTERFACES=hme0 hme1

NODE=10 08:00:20:f9:c5:54
NODE=20 08:00:20:f9:a8:12

SLICE=c0t0d0s0 2048 /               -        logging
SLICE=c0t0d0s1 1024 swap            -        -
SLICE=c0t0d0s3 3    unnamed         -        -
```

EXAMPLE 2 Example of a configuration for a cluster with no diskless nodes *(Continued)*

```
SLICE=c0t0d0s4 3    unnamed         -        -
SLICE=c0t0d0s5 2048 /SUNWcgha/local c0t0d0s3 logging
SLICE=c0t0d0s6 2048 /user1          c0t0d0s4 logging
SLICE=c0t0d0s7 free /test1          -        logging

EXPORTED=/user1/export

MOUNTED=/user1_app /user1/export/app
```

EXAMPLE 3 Example of a configuration for a cluster with volume-managed, replicated partitions for FC-AL disks

```
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
#
#
# The usage of shell variables is authorized.
# This file is sourced by the nhinstall script.
#

#-----------------------------------------------------
#
#  This file enables you to define the cluster
#  environment.
#
#-----------------------------------------------------

CLUSTER_ID=250

ARCH=sun4u

MEN_INTERFACES=hme0 hme1

NMEN_INTERFACES=eri0 eri1

LOGICAL_SLICE_SUPPORT=YES

NODE=10 08:00:20:f9:c5:54
NODE=20 08:00:20:f9:a8:12
NODE=30 -                       - node30

SLICE=c0t0d0s0 2048 /               -        logging
SLICE=c0t0d0s1 1024 swap            -        -
SLICE=c0t0d0s3 2048 /export         c0t0d0s5 logging
SLICE=c0t0d0s4 2048 /SUNWcgha/local c0t0d0s6 logging
SLICE=c0t0d0s5 3    unnamed         -        -
SLICE=c0t0d0s6 3    unnamed         -        -
SLICE=c0t0d0s7 free replica         -        logging
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhins

SEE ALSO

`nhinstall`(1M), `nhadm`(1M), `nhenablesync`(1M), `addon.conf`(4), `env_installation.conf`(4), `hosts`(4), *Solaris Installation Guide*

NAME

cluster_nodes_table – central cluster management file

SYNOPSIS

/etc/opt/SUNWcgha/cluster_nodes_table

DESCRIPTION

The `cluster_nodes_table` contains a definition for each node in the cluster. The `cluster_nodes_table` is located on each master-eligible node. This file stores the membership and configuration information for all peer nodes in a cluster including potential future nodes. A node must have an entry in the `cluster_nodes_table` to be part of a cluster.

If you install the software on the cluster manually, as described in the *Netra High Availability Suite Foundation Services 2.1 6/03 Custom Installation Guide*, you must create this file on each master-eligible node. A template of the `cluster_nodes_table` file is available in `/etc/opt/SUNWcgha/cluster_nodes_table.template`.

By default, the `cluster_nodes_table` is located in the `/etc/opt/SUNWcgha/` directory on each master-eligible node. You can change the path to the cluster node table by editing the parameter `CMM.LocalConfig.Dir` parameter in the `nhfs.conf` file. For more information on `nhfs.conf`, see the `nhfs.conf`(4) man page.

The following is an example of a cluster node table with three peer nodes:

```
#NodeId       Domain_id     Name                    Attributes
10            250           netraMEN1-cgtp0         -
20            250           netraMEN2-cgtp0         –
30            250           netraDISKLESS1-cgtp0    –
```

`NodeId` This is the unique node ID within the cluster. This number is the decimal equivalent of the host part of the node's IP address.

`Domain_id` This is the unique cluster ID within the cluster domain and must be the same for each peer node. This `Domain_id` must be the same as the one defined in the `nhfs.conf` file. For more information see the `nhfs.conf`(4) man page.

`Name` This is the node name. The name must be the same as the one in the `nhfs.conf` file on the node. See the `nhfs.conf`(4) man page.

`Attributes` Each node can be assigned the following attributes:

Note – Do not alter the role of a node by modifying the attribute column of `cluster_nodes_table` file.

`D` Disqualified

`D` corresponds to the `CMM_DISQUALIFIED_MEMBER` attribute of the CMM API. If a node is flagged as disqualified, it cannot be assigned a master or vice-master role. This attribute is only applied to a master-eligible node.

`S` Synchronization needed

If a node has the `S` flag, the disks of the master node and vice-master nodes are not synchronized. This is a read-only flag and must not be changed manually. This attribute applies only to master-eligible nodes.

`-` Not Disqualified

If a node has this attribute, the node is not disqualified and is in the cluster.

To check the role of a node, use the `nhcmmstat` command, as described in the `nhcmmstat`(1M) man page.

EXTENDED DESCRIPTION

If you want to add a new node to a cluster you must either verify that the node already has an entry in the cluster node table or create an entry for this node in the table. You must only modify this file during initial cluster configuration if you install the software manually on the cluster or if you are adding or removing a node. For more information, see the `cmm_config_reload`(3CMM) man page.

When editing the cluster node table file, make sure that:

1. The `NodeId` for each node in the `cluster_nodes_table` is unique. If this is not the case, use `nhcmmstat` to determine the correct `NodeId` of the peer nodes and modify the cluster node table.
2. The `NodeId` for each node in the `cluster_nodes_table` has a value *n* such that 3<*n*<255. If this is not the case, modify the cluster node table.
3. The master-eligible nodes have the attribute "`-`" in `cluster_nodes_table`. If this is not the case, correct this error in the file.

All other updates are performed automatically by the `nhcmmd` daemon. The `nhcmmd` daemon on the master node uses the cluster node table to know which nodes are in the cluster and what attributes they are assigned.

The `cluster_nodes_table` is read and changed automatically by the `nhcmmd` daemon as follows:

1. When the master node is elected:
   a. The master node reads the `cluster_nodes_table` and stores it in memory.
   b. The master node broadcasts this view of the `cluster_nodes_table` to peer nodes in the cluster.

c. Each master-eligible node receives this view and stores this view of the `cluster_nodes_table`.

2. When the cluster is running:

   a. After the cluster is up and running, the `cluster_nodes_table` is only read when a `cmm_config_reload()` is called or when the following command is executed:

      ```
      # nhcmmstat -c reload
      ```

   b. The master node reads and stores the new view of the `cluster_nodes_table` in memory.

   c. The master node broadcasts the new view to peer nodes in the cluster.

   d. The vice-master node receives this view and stores this view of the `cluster_nodes_table`.

3. The `cluster_nodes_table` is modified when an attribute of a node has been changed through the API, for example, `cmm_member_setqualif()`.

The preceding methods of creating and managing the `cluster_nodes_table` file, ensures that the `cluster_nodes_table` is *persistent*, that is, if the cluster goes down for any reason, the same `cluster_nodes_table` file is restored when the cluster comes back up, no matter which master-eligible nodes is elected master.

WARNINGS

Changes should not be made to the cluster node table without careful consideration because the cluster node table maintains the central view of the cluster membership and node status. Altering the cluster node table can result in a cluster that is not highly available.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcma
Interface Stability	Evolving

SEE ALSO

`nhcmmd`(1M), `nhadm`(1M), `nhfs.conf`(4), `nhcmmstat`(1M), `cmm_config_reload`(3CMM)

NAME

diskless_nodeprof.conf – `permit the customization of Solaris installation on diskless nodes`

SYNOPSIS

`diskless_nodeprof.conf`

DESCRIPTION

Creating the `diskless_nodeprof.conf` file enables you to to install a different Solaris distribution in the diskless node environment of a cluster to the distribution you are installing on the master-eligible nodes. To use the `diskless_nodeprof.conf` file you must be installing a cluster with `nhinstall`.

When the `diskless_nodeprof.conf` file is present in the local directory on the installation server, the file's contents supersede the contents of both the `nodeprof.conf` file and the default profile file that `nhinstall` uses for diskless node installation. If the `diskless_nodeprof.conf` file exists, `nhinstall` uses the `nodeprof.conf` file for details of the Solaris profile it is to install on the master-eligible nodes and the `diskless_nodeprof.conf` file for the Solaris profile it is to install in the diskless node environment.

Note – If you want to install the same Solaris profile on master-eligible and diskless nodes, do not create a `diskless_nodeprof.conf` file.

The first reference to *cluster* in the `diskless_nodeprof.conf` file must refer to the *cluster* parameter to be given to the `smosservice add` command. Any other directives given in this file will be ignored. For more information about the `smosservice` command, see the `smosservice`(1M) man page.

Note – You cannot use `nhinstall` to install software on dataless nodes.

For information about the format and contents of the `diskless_nodeprof.conf` file, see the `nodeprof.conf` file and "Preparing Custom JumpStart Installations (Tasks)" in the *Solaris Installation Guide*.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC

SEE ALSO

`nhinstall`(1M), `smosservice`(1M), `nodeprof.conf`(4), *Solaris Installation Guide*

NAME

env_installation.conf – `nhinstall` configuration file defining the installation environment

SYNOPSIS

`env_installation.conf`

DESCRIPTION

Configure the `env_installation.conf` file to define the installation environment for the `nhinstall` tool.

The templates for the configuration files are contained in the `/opt/SUNWcgha/config.standard` directory with `.template` extensions. Copy the configuration files to a local directory on the installation server as follows:

```
# mkdir config-file-directory
# export NHOME=/opt/SUNWcgha/config.standard
# cd config-file-directory
# cp $NHOME/env_installation.conf.template env_installation.conf
```

Note – All the configuration files must be in the same local directory on the installation server.

The `env_installation.conf` file format is ASCII. Comment lines begin with the comment mark (#). Parameters consist of a keyword followed by an equals (=) sign followed by the parameter value, of the form:

Keyword=Value

The following *Keyword* and *Value* parameters are supported:

EXTENDED DESCRIPTION

Modify variables in the `env_installation.conf` file as follows:

`SERVER_INTERFACE`
The network interface of the installation server used to access the cluster when installing the product, for example:

```
SERVER_INTERFACE=hme1
```

`SERVER_NODE`
The node ID of the installation server within the cluster. This must be a unique value between 3 and 254, for example:

```
SERVER_NODE=253
```

`SERVER_IP`
The IP address of the installation server. This IP address is required only when a public network is used, that is, the `PUBLIC_MEN_INTERFACES` variable is defined in `cluster_definition.conf`. If you do not specify the `SERVER_IP` variable, an IP address is automatically created based on the cluster ID, `CLUSTER_ID`, and the node ID, `SERVER_NODE`. The `CLUSTER_ID` is specified in `cluster_definition.conf`.

```
SERVER_IP=192.168.12.50
```

Note – If you plan to configure external IP addresses for nodes in the cluster by configuring the PUBLIC_NETWORK parameter in the cluster_definition.conf file, make sure that the IP address specified in the SERVER_IP parameter is on the same subnet as the one specified in the PUBLIC_NETWORK parameter. The PUBLIC_NETWORK parameter configures the network interface of the installation server.

AUTO_REBOOT
When launched, the nhinstall tool automatically reboots the nodes as required during the installation process.

Options are YES and NO. The default is NO.

SOLARIS_INSTALL
Install the Solaris operating system. Options are ALL, NONE, and DISKLESS_ONLY. The default is ALL.

If you choose NONE, the Solaris operating system is not installed on the cluster nodes. In this case, make sure of the following before launching the nhinstall tool:

- The Solaris operating system is already installed on the nodes.
- Java Development Kit (JDK) is not already installed on the cluster nodes.

If you choose DISKLESS_ONLY, the Solaris operating system is not installed on the master-eligible nodes. You must make sure that the Solaris operating system is already installed on the nodes before running the nhinstall tool. When you launch the nhinstall tool, the Solaris services for the diskless nodes are installed.

If you are using Fibre Channel-Arbitrated Loop (FC-AL) disks on Netra 20 servers, see "Configuring the nhinstall Tool" in *Netra High Availability Suite Foundation Services 2.1 6/03 Custom Installation Guide* before setting this parameter.

NHAS2_PRODUCT_DIR
The directory containing the Foundation Services distribution, for example:

```
NHAS2_PRODUCT_DIR=/cdrom
```

WORKING_DIR
The directory on the installation server where temporary files are created during the installation process. This directory must be writable and shared. The progress indicator used for installation recovery is also stored in this directory. Do not specify the /tmp directory, which is deleted in the event of a reboot.

```
WORKING_DIR=/export/nhtmp
```

SOLARIS_DIR
The location of the Solaris distribution, for example:

```
SOLARIS_DIR=/export/su28u7fcs
```

DISKLESS_SOLARIS_DIR
The location of the Solaris distribution for diskless nodes, if this distritbution is not the same as that installed on the other nodes in the cluster. By default, the Solaris distribution installed on the master-eligible nodes is used for DISKLESS_SOLARIS_DIR.

```
DISKLESS_SOLARIS_DIR=$SOLARIS_DIR
```

EXAMPLES

EXAMPLE 1 Sample env_installation.conf File

```
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
#
# The usage of shell variables is authorized.
# This file is sourced by the nhinstall script.
#-------------------------------------------------
#
#  This file enables you to define the installation
#  environment.
#
#-------------------------------------------------
#
# Installation server network interface used to access the cluster.
#
SERVER_INTERFACE=hme1

#
# The node id attributed to the installation server within the cluster.
#
SERVER_NODE=253

#
#  The public IP address used only when PUBLIC_MEN_INTERFACES is configured.
#
#  For example:
#      SERVER_IP=192.168.12.50
#

#
# Install Solaris on the master-eligible nodes.
# Options are ALL, NONE, or DISKLESS_ONLY. The default is ALL.
SOLARIS_INSTALL=ALL

#
# Automatically reboot the master-eligible nodes after:
#  . The Foundation Services are installed
#  . The diskless environment is configured
#
# Options are YES or NO. The default is NO.
#
AUTO_REBOOT=NO

#
# IMPORTANT:
#  All directories mentioned below will be exported (via the share command)
#  Please check that they can be exported. The directories must not be the
#  child or parent of a directory already exported.
```

EXAMPLE 1 Sample `env_installation.conf` File *(Continued)*

```
#

#
# Location of the Foundation Services software distribution.
# (usually the CDROM reader)
#
NHAS2_PRODUCT_DIR=/cdrom

#
# Define the working directory where:
# . The progress indicator file is stored.
# . The JumpStart directory and files used for installing Solaris
#   on nodes are created.
# This directory must be writable.
#
WORKING_DIR=/export/nhtmp

#
# Location of the Solaris distribution on the installation server.
#
SOLARIS_DIR=/export/home/s9hwl3
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhins

SEE ALSO

`nhinstall`(1M), `cluster_definition.conf`(4), `addon.conf`(4)

NAME

install-server.conf – SMCT configuration file to configure the network for the installation server

SYNOPSIS

SMCT_DEFAULT_CONFIG_DIR/models/install-server.conf

*config-dir***/models/install-server.conf**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `install-server.conf` configuration file enables network configuration between the prototype machine and the installation server. The prototype machine and the installation server must be on the same subnet.

PARAMETERS

The following are the parameters in the `install-server.conf` file:

```
# install-server network configuration file
name install-server-name
ip install-server-IP
netmask netmask
nic nic-name
router y/n
```

- *install-server-name*

  Host name of the installation server.

- *install-server-IP*

  IP address of the installation server.

- *netmask*

  Netmask of the private network between the installation server and the prototype machine.

- *nic-name*

  Interface name of the installation server's private network.

- *y/n*

  By default, SMCT verifies that the installation server does not act as a router, that is, the file `/etc/notrouter` exists. This avoids the broadcasting of routing information from the private network over the public network. To enable routing, set the parameter to `y`. The default value is `n`.

The `name`, `ip`, `netmask`, and `nic` parameters are mandatory and must be configured. The `router` parameter is optional.

EXAMPLE

The following is an example of an `install-server.conf` file:

```
# install-server network configuration file
name installserv1
ip 10.100.1.1
netmask 255.255.255.0
nic hme1
router n
```

ATTRIBUTES See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO `flcreate`(1M), `fldeploy`(1M), `flinstall`(1M), `master-system.conf`(4)

NAME

machine.conf – SMCT configuration file to define the cluster in terms of hardware elements, disk layout, and file system

SYNOPSIS

SMCT_CONFIG_DIR/models/machine.conf

*smct-config-dir***/models/machine.conf**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `machine.conf` configuration file enables you to define the hardware components, disk layout for master-eligible and dataless nodes, and the file system of the cluster.

A pre-configured `machine.conf` template file for each example hardware configuration is available in the `/opt/SUNWcgha/nhsmct/etc/models/` directory. For a description of the files in this directory, see the *Netra High Availability Suite Foundation Services 2.1 6/03 README*.

The `machine.conf` configuration file contains the following sections:

- Cluster hardware definition

  This section is a hierarchical description of the cluster hardware, starting with a top level description of the contents of the shelf, which typically contains drawers and switches. The cluster hardware definition is referenced by the cluster definition in the `cluster.conf` file.

  The following configuration elements are defined by the `ELEMENT` keyword:

  - `drawer`

    A container that holds one or several boards and disks.

  - `board`

    The CPU board associated to a node and the network interface cards (NIC) that it uses. Nodes are defined in the `cluster.conf` file.

  - `switch`

    The set of ports used to connect the NICs.

  - `port`

    The connection between the switch ports and the NICs.

- Disk layout

  The disk layout section contains a description of the disk layout for the master-eligible nodes and dataless nodes.

  The following configuration elements are defined by the `ELEMENT` keyword:

  - `disk`

    A disk is described in terms of slices by using the `slice` configuration element.

  - `slice`

Each slice is described in terms of size of partition, the file system to be mapped to the slice, and whether the slice is replicated or not, using the attributes `number`, `size`, `type`, and `role`.

- File system definitions

  The file system definition section contains definitions of the file systems on each disk. The following configuration element is defined by the `ELEMENT` keyword:

  - `filesys`

    Each file system is defined in terms of root directories, read and write permissions, if the file system is exported, and if the file system is shared.

- `link`

  A direct link connecting the serial ports of the master-eligible nodes can be configured to prevent a split brain situation, which is an error scenario where a cluster has two master nodes.

The `ip` configuration elements referenced in the `machine.conf` file are defined in the `network.conf` file.

PARAMETERS

This section describes the parameters in the `machine.conf` file:

```
{ ELEMENT shelf name
                  CONTAIN {drawer name}+ {switch name}*
}+

{ ELEMENT drawer name type drawer-type
                  [ CONTAIN {board name}* {disk name}* ]
}+

{ ELEMENT drawer name type drawer-type
    [ CONTAIN {board name}* {disk name}* ]
}+

{ ELEMENT switch name type switch-type
    [ CONTAIN  {port name}* ]
    [ USE {ip name}* ]
}*

{ ELEMENT port number number number-value
    CONNECT nic name
}*

{ ELEMENT board name arch arch-type type board-type
    [ vendorType board-vendorType ] [ class board-class ]
 [ clientId board-clientId ]
    [ USE {nic name}+ [link name]
}+

{ ELEMENT nic name device nic-device type nic-type role nic-role
    class nic-class [address nic-address]
    [ USE ip name ]
}*

{ ELEMENT disk name device disk-device type disk-type size disk-size
    CONTAIN {slice name}+
```

```
}*

{ ELEMENT slice name number slice-number type slice-type size slice-size
    rawDev rawDev-name blockDev blockDev-name [role slice-role]
    [ MAP filesys name ]
    [ USE slice name ]
    [ MANAGE replicatedSlice name ]
}*

{ ELEMENT filesys name role filesys-role type filesys-type
    [ size filesys-size ] [ fsck filesys-fsck ] [ mntPt filesys-mntPt ]
    [ mntBt y/n ] [ mntOpt filesys-mntOpt ] [ remMntPt filesys-mntOpt ]
    [ CONTAIN {exportedFileSys name}+ ]
}*

{ ELEMENT link name device name [ speed speed-value ] } *
```

- *name*

  The unique name of the configuration record.

- *drawer-type*

  The type of drawer. This can be one of the following:

  - CT410
  - CT810
  - CT821
  - NETRA_20
  - NETRA_120
  - T1_105
  - T1_200

- *switch-type*

  The type of switch. This must be set to GENERIC.

- *board-type*

  The type of board. This can be one of the following:

  - CP2140
  - CP2160
  - CP2300
  - T4
  - V120
  - T1_105
  - T1_200
  - GENERIC
  - sun4u

- *board-class*

  The type of machine hardware, for example, sun4u. Configure this parameter if you have set the *board-type* to GENERIC. It is not necessary to configure this parameter for non-generic boards as it is pre-configured with the output from the uname -m command.

- *board-vendorType*

The type of hardware implementation. For example, `SUNW,UltraSPARC-IIi-cEngine`. Configure this parameter if you have set the *board-type* to `GENERIC`. It is not necessary to configure this parameter for non-generic boards as it is pre-configured with the output from the `uname -i` command.

- *disk-size*

  The size of the disk in Gbytes.

- *arch-type*

  The type of architecture. This must be set to `SPARC`.

- *nic-type*

  The type of NIC. This can be one of the following:

  - `PHYSICAL`

    For a NIC that is not aliased. For example, `eri0`.

  - `VIRTUAL`

    For a NIC that is not related to a physical device. For example, `cgtp0`.

  - `LOGICAL`
  - For a NIC that is aliased on a physical NIC. For example, `eri1:10`.

- *nic-class*

  The class of the NIC. This can be one of the following:

  - `PRIMARY`

    For a NIC using the Solaris JumpStart configuration.

  - `SECONDARY`

    For a NIC that is not using the Solaris JumpStart configuration.

- *nic-role*

  The role of the NIC. A NIC can be assigned to the CGTP or the Node State Manager (NSM).

  For the CGTP service, you need to choose three NICs:

  - A NIC associated to physical subnet A with the role `CGTP_NIC0`.
  - A NIC associated to physical subnet B with the role `CGTP_NIC1`.
  - A NIC associated to the virtual subnet CGTP with the role `CGTP`.

  For the NSM service, you need to choose a NIC associated to an external subnet with the role `NSM`.

- *nic-address*

  The Ethernet address of the NIC. This parameter is needed for:

  - Master-eligible nodes and dataless nodes. This parameter must be configured for the `PRIMARY` NIC to configure Solaris JumpStart.

- Diskless nodes. This parameter must be configured for the `PRIMARY` and `SECONDARY` NICs if the boot policy of the associated node group set to `MAC_ADDR_POLICY` in the `cluster.conf` file.

- *disk-type*

  The type of the disk. This can be one of the following:

  - `SCSI`
  - `IDE`
  - `FC` (Fibre Channel-Arbitrated Loop)

- *slice-number*

  The *slice-number* parameter is used with the disk device name to define the name of the slice.

  The name of the slice = *disk-device*+ *slice-number*.

  For example:

  ```
  c0t0d0s0 = c0t0d0 + s0
  ```

  In this case, the *slice-number* is `0`.

- *slice-rawDev*

  The name of the raw-device family associated to the slice. For example, `/dev/rdsk`.

- *slice-blockDev*

  The name of the block-device family associated to the slice. For example, `/dev/md/dsk`.

- *slice-type*

  The type of slice. This can be one of the following:

  - `PHYS`

    A slice that is not managed by Solaris Volume Manager.

  - `META`

    A slice managed by Solaris Volume Manager that is the same size as the related `PHYS` slice

  - `SOFT`

    A slice managed by Solaris Volume Manager that is less than or equal to the related `PHYS` slice.

  The `PHYS` and `META` slices are associated to `PHYS` slices by the keyword `USE`.

- *slice-role*

  The role of the slice. This can be one of the following:

  - `META_DB`

    Used for Solaris Volume Management.

  - `BITMAP_SNDR`

Used by Sun StorEdge Network Data Replicator (SNDR) for each `REPLICATED` slice. `BITMAP_SNDR` slices can be either `PHYS` slices or `SOFT` slices.

- `REPLICATED`

  Slices that SNDR must manage. `REPLICATED` slices can either be `PHYS` slices or `META` slices.

If a `REPLICATED` slice is defined, then you must define a `BITMAP_SNDR` slice, linked to the previous one by the keyword `MANAGE`.

- *slice-size*

  The number of cylinders multiplied by 1 Mbyte.

- *y/n*

  This is yes or no for the `mntBt` option, that is, the mount at boot option.

- *filesys-role*

  The role of the file system. This can be one of the following:

  - `root`

    Defines the root file system.

  - `swap`

    Defines the swap file system.

  - `export`

    Defines the file system containing the diskless environment.

  - `shared`

    Defines the file system that contains the software shared by all the cluster nodes.

  - `user`

    User-defined file system

  - `data`

    NFS file system exporting cluster-wide data.

  - `database`

    NFS file system exporting shared software database.

  - `services`

    NFS file system exporting shared software.

  The `data`, `database`, and `services` file systems are related to the `shared` file system.

- *filesys-type*

  The type of file system. This can be one of the following:

  - `ufs`
  - `nfs`
  - `swap`

- *filesys-size*

  The size of the file system in Mbytes.

- *filesys-fsck*

  The file system to be checked. This can be either 1 or 2.

- *filesys-mntPt*

  The mount point for the file system. The value can include the `<--SWLID-->` tag, which is replaced with the software load version by the SMCT.

- *filesys-mntOpt*

  The mount options for the file system.

- *filesys-remMntPt*

  The exported mount point for the shared file systems.

- *speed-value*

  The speed of the serial line. Valid values are 38400, 57600, 76800, and 115200. The default value is 115200.

EXAMPLES Following is an example of the disk layout section of the `machine.conf` file:

```
ELEMENT slice s0@disk1 number 0 rawDev /dev/rdsk blockDev /dev/dsk
        type PHYS size 2048
        MAP filesys ROOT

# Swap partition
ELEMENT slice s1@disk1 number 1 rawDev /dev/rdsk blockDev /dev/dsk
        type PHYS size 1024
        MAP filesys SWAP

# Shared software and data partition (replicated)
ELEMENT slice s3@disk1 number 3 rawDev /dev/rdsk blockDev /dev/dsk
        type PHYS size 2048

ELEMENT slice d3@disk1 number 3 rawDev /dev/md/rdsk blockDev /dev/md/dsk
        type META size 2048 role REPLICATED
        USE slice s3@disk1
        MAP filesys SHARED

ELEMENT slice d31@disk1 number 31 rawDev /dev/md/rdsk blockDev /dev/md/dsk
        type SOFT size 10 role BITMAP_SNDR
        USE slice s7@disk1
        MANAGE replicatedSlice d3@disk1

# User partition 1 (replicated)
ELEMENT slice s4@disk1 number 4 rawDev /dev/rdsk blockDev /dev/dsk
        type PHYS size 2048

ELEMENT slice d4@disk1 number 4 rawDev /dev/md/rdsk blockDev /dev/md/dsk
        type META size 2048 role REPLICATED
        USE slice s4@disk1
        MAP filesys USER1
...
```

The following is an example of a configuration for the direct link via a serial line between the two master-eligible nodes:

```
ELEMENT link serial-b device /dev/term/b speed 57600

ELEMENT board T1105@peerNode1 type T1_105 arch SPARC
         USE nic nic0@peerNode1
             nic nic1@peerNode1
             nic cgtp@peerNode1
             link serial-b
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`cluster.conf`(4), `network.conf`(4), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M)

NAME

master-system.conf – SMCT network configuation file to connect the prototype machine and the installation server

SYNOPSIS

`SMCT_CONFIG_DIR/models/master-system.conf`

config-dir`/models/master-system.conf`

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The `master-system.conf` configuration file enables network configuration between the prototype machine and the installation server.

PARAMETERS

The `master-system.conf` file contains the following parameters:

```
# master system network configuration file
name prototype-machine-name
ip prototype-machine-IP
mac prototype-machine-ethernet-address
```

- *prototype-machine-name*

  Host name of the prototype machine.

- *prototype-machine-IP*

  IP address of the prototype machine.

- *prototype-machine-ethernet-address*

  Ethernet address of the prototype machine. Do not define this parameter if *prototype-machine-name* is already defined in the file `/etc/ethers`.

EXAMPLE

The following is an example of a `master-system.conf` file:

```
# master system network configuration file
name protomachine1
ip 10.100.1.10
netmask 255.255.255.0
mac 8:0:20:f9:c2:b0
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`flcreate`(1M), `flinstall`(1M), `install-server.conf`(4)

NAME

network.conf – SMCT configuration file to define the network parameters for the cluster

SYNOPSIS

SMCT_CONFIG_DIR/models/network.conf

*smct-config-dir***/models/network.conf**

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

Configure the `network.conf` file to define specific network information used by the target cluster. This includes IP addresses, network, and netmask information for physical and virtual cluster network interfaces, and default routes used to access the cluster.

A pre-configured `network.conf` template file for each example hardware configuration is available in the `/opt/SUNWcgha/nhsmct/etc/models/` directory.

PARAMETERS

The following are the parameters of the `network.conf` file:

```
{ ELEMENT network name type network-type number network-number
netmask network-netmask
}+

{ ELEMENT ip name address ip-address type ip-type {alias name}*
    BELONG_TO network name}+

{ ELEMENT router name
    USE ip name}+
```

- *name*

  Unique ID as an ASCII string.

- *network-type*

  Network type. This parameter must be set to `IPV4`.

- *network-number*

  IP network number in standard IP network number format (dotted notation), for example, `10.101.1.0`.

- *network-netmask*

  IP network netmask in standard IP network netmask format (dotted notation), for example, `255.255.255.0`.

- *ip-type*

  IP address type. This can be one of the following:

  - `FLOATING`

    The floating address triplet of the master node referenced by the configuration element `domain` in `cluster.conf` file. Alternatively, you can specify the floating address used by the Node State Manager, which is defined in the `cluster.conf` file.

  - `STATIC`

All IP addresses that are not of type `FLOATING`.

- *ip-address*

  IP address in standard IP address format (dotted notation), for example, `10.101.1.10`.

  You can choose to configure IP addresses of any class for your nodes. However, the following rules apply:

  - The *nodeid* in the `cluster.conf` file must be a decimal representation of the host part of the corresponding IP address.
  - The host part of the IP address for the *NIC0*, *NIC1*, and `cgtp0` interfaces of a node must be the same.

EXAMPLES The following is an example of a `network.conf` file for a four-node cluster:

```
# ----------------------------------------------------------
#   ident "@(#)network.conf.4N.tmpl 1.5"
# ----------------------------------------------------------------
#      - 4 nodes reference platform network definition template -
# ----------------------------------------------------------------
#
ELEMENT network phys-A type IPV4 number 10.101.1.0 netmask 255.255.255.0
ELEMENT network phys-B type IPV4 number 10.101.2.0 netmask 255.255.255.0
ELEMENT network cgtp type IPV4 number 10.101.3.0 netmask 255.255.255.0
#
# Cluster nodes IP addresses
#
ELEMENT ip peerNode1-4N-nic0 address 10.101.1.10 type STATIC alias node1
        BELONG_TO network phys-A
ELEMENT ip peerNode1-4N-nic1 address 10.101.2.10 type STATIC
        BELONG_TO network phys-B
ELEMENT ip peerNode2-4N-nic0 address 10.101.1.20 type STATIC alias node2
        BELONG_TO network phys-A
ELEMENT ip peerNode2-4N-nic1 address 10.101.2.20 type STATIC
        BELONG_TO network phys-B
ELEMENT ip peerNode3-4N-nic0 address 10.101.1.30 type STATIC alias node3
        BELONG_TO network phys-A
ELEMENT ip peerNode3-4N-nic1 address 10.101.2.30 type STATIC
        BELONG_TO network phys-B
ELEMENT ip peerNode4-4N-nic0 address 10.101.1.40 type STATIC alias node4
        BELONG_TO network phys-A
ELEMENT ip peerNode4-4N-nic1 address 10.101.2.40 type STATIC
        BELONG_TO network phys-B
#
# Master floating addresses
#
ELEMENT ip master-cgtp address 10.101.3.1 type FLOATING alias master
        BELONG_TO network cgtp

ELEMENT ip master-nic0 address 10.101.1.1 type FLOATING
        BELONG_TO network phys-A

ELEMENT ip master-nic1 address 10.101.2.1 type FLOATING
        BELONG_TO network phys-B

#
```

```
# Cluster nodes CGTP addresses
#
ELEMENT ip peerNode1-4N-cgtp alias peerNode1-4N-cgtp.localdomain
        address 10.101.3.10 type STATIC
        BELONG_TO network cgtp
ELEMENT ip peerNode2-4N-cgtp alias peerNode2-4N-cgtp.localdomain
        address 10.101.3.20 type STATIC
        BELONG_TO network cgtp
ELEMENT ip peerNode3-4N-cgtp alias peerNode3-4N-cgtp.localdomain
        address 10.101.3.30 type STATIC
        BELONG_TO network cgtp
ELEMENT ip peerNode4-4N-cgtp alias peerNode4-4N-cgtp.localdomain
        address 10.101.3.40 type STATIC
        BELONG_TO network cgtp
#
# Router ip addresses
#
ELEMENT ip router-4N-nic0 address 10.101.1.90 type STATIC
        BELONG_TO network phys-A
ELEMENT ip router-4N-nic1 address 10.101.2.90 type STATIC
        BELONG_TO network phys-B
#
# Default router configuration
#
ELEMENT router default-router
        USE ip router-4N-nic0
            ip router-4N-nic1
#
# External access to switches
#
ELEMENT ip switch1-4N-nic0 address 10.101.1.100 type STATIC
        BELONG_TO network phys-A
ELEMENT ip switch2-4N-nic0 address 10.101.2.200 type STATIC
        BELONG_TO network phys-B
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`cluster.conf`(4), `machine.conf`(4), `slconfig`(1M), `slcreate`(1M), `sldelete`(1M), `sldeploy`(1M), `slexport`(1M)

NAME

nhadmsync.conf – list of nonreplicated files and the differences between them

SYNOPSIS

/SUNWcgha/remote/etc/nhadmsync.conf

DESCRIPTION

The nhadmsync.conf file is the configuration file for the nhadm synccheck and nhadm syncgen commands. The nhadm synccheck command compares nonreplicated files on the master and the vice-master nodes, printing any differences to the console. You can accept the differences using the nhadm syncgen command.

To use these commands, both master-eligible nodes must have remote access to the other master-eligible node. For details on how to enable this, see the nhadm(1M) man page.

EXTENDED DESCRIPTION

In the nhadmsync.conf, you must specify the nonreplicated files that you want to compare. The default location of this file is /SUNWcgha/remote/etc/nhadmsync.conf. The name and location of this file can be changed at any time and is specified in the -y | –syncfile option, when using the nhadm synccheck command.

You can create multiple versions of the nhadmsync.conf file. This enables you to have lists that are specific to a feature or a group of features, as described in the nhadm(1M) man page. The nhadm synccheck configuration files must have write permissions if you want to use the nhadm syncgen command.

To use the nhadmsync.conf file, copy the template file /opt/SUNWcgha/config.standard/adm/nhadmsync.conf.template to /SUNWcgha/remote/etc/nhadmsync.conf.

Add the names of the files to be compared, to the nhadmsync.conf file. Make sure that the filenames you add have the following criteria:

- The files exist on both master-eligible nodes
- The files are not replicated on a shared file system

The syntax of your entries in this file must be the following:

```
NODEID=node1  node2
FILE=filename1
 =BEGIN
 ...
  =END
FILE=filename2
 =BEGIN
 ...
  =END
```

NODEID

node1 and *node2* are the node IDs of the master and vice-master nodes, respectively. If these are not present, the default logical IP addresses of the master and vice-master nodes are used.

When nhadm syncgen is executed, the NODEID parameter is generated with the actual node IDs of both nodes to ensure the comparison is always made in the same order. This is because the diff -b command is dependent on the order of the files.

If the NODEID parameter is present, it must be the first line of the nhadmsync.conf file and can only be preceded by a blank line.

FILE
The name of the file to be tested.

=BEGIN...=END
This contains the result of the diff -b command for the file specified by the preceding FILE parameter.

EXAMPLES

EXAMPLE 1 Example of information added by syncgen after the file comparisons

If you defined the nhadmsync.conf file as follows:

```
NODEID=10 20
FILE=/etc/ethers
FILE=/etc/hosts
FILE=/etc/netmasks
```

After the nhadm syncgen command is executed, the nhadmsync.conf file might contain the following information:

```
NODEID=10 20
FILE=/etc/ethers
FILE=/etc/hosts
=BEGIN
5c5,6
< 10.250.1.10   MEN-C250-N10    loghost
---
> 10.250.1.20   MEN-C250-N20    loghost
> 10.250.1.10 MEN-C250-N10
8d8
< 10.250.1.20 MEN-C250-N20
=END
FILE=/etc/netmasks
```

The differences printed to the nhadmsync.conf file are the differences that would be found by running diff -b on the files listed in nhadmsync.conf. For more information on the diff command, see the diff(1) man page.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhadm
Interface Stability	Evolving

SEE ALSO

nhadm(1M), diff(1)

NAME

nhfs.conf – Foundation Services configuration file

SYNOPSIS

`/etc/opt/SUNWcgha/nhfs.conf`

DESCRIPTION

The `nhfs.conf` file contains configuration information for the Foundation Services such as the Cluster Membership Manager, Reliable NFS, and the Node State Manager. This file also provides cluster addressing and interface configuration information.

Configure this file on each node if you plan to install the Foundation Services manually. To manually configure the parameters in the `nhfs.conf` file, uncomment the parameters, that is, delete the comment mark (#) at the beginning of the line, and modify the value of the parameter. A description of each parameter is provided in the following sections.

When you have manually install the Foundation Services packages, copy the template file `/etc/opt/SUNWcgha/nhfs.conf.template` to the default location `/etc/opt/SUNWcgha` as `nhfs.conf` on each peer node. For each file, make the necessary modifications in a text editor.

Do not re-edit the `nhfs.conf` file when the cluster is running.

The `nhfs.conf` file format is ASCII. Parameters consist of a keyword followed by an equals (=) sign followed by the parameter value, of the form:

Keyword=Value

The following *Keyword* and *Value* parameters are supported.

COMMON PARAMETERS

`Cluster.DataManagementPolicy`

Define how the cluster behaves when the vice-master node starts up while the master node is down. You can select one of three data management policies. Values are `Integrity`, `Availability`, and `Adaptability`. The default value is `Integrity`.

```
Cluster.DataManagementPolicy=Integrity | Availability | Adaptability
```

`Integrity`	If there is a vice-master but no master in the cluster, the vice-master waits for the old master to rejoin the cluster before it takes the master role. This ensures that the cluster has the most up-to-date data. Choosing this value might reduce the availability of the cluster, but it prioritizes data integrity.
`Availability`	If there is a vice-master in the cluster but no master, the vice-master does not wait for the old master to rejoin the cluster before taking on the master role. The previous master is still off-line. Any data that is written to the new master while there is no vice-master will be lost. When the old master node comes back on line as the vice-master node, a full synchronization occurs between the two master-eligible nodes.

`Adaptability`
: If there is a vice-master in the cluster but no master, the vice-master checks the disk synchronization state. If the state is not synchronized, that is the state returned by `nhcmmstat` is `synchro:NEEDED`, the vice-master waits for the master to come up. This is equivalent to the `Integrity` data management policy. If the state is synchronized, that is the state returned by `nhcmmstat` is `synchro:READY`, the vice-master is elected the new master. This is equivalent to the `Availability` data management policy.

`Cluster.Master.ID`
: Specify the host part of the master node's floating IP address in decimal form. The floating address triplet for the node with the master role is calculated from the local network interface addresses, the netmask, and the value of this parameter. These addresses are IPv4 address.

  The default value is `1`.

```
Cluster.Master.ID=1
```

  If the `Cluster.Master.ID` is `1` and the cluster network is set up as follows:

  - Netmask: `255.255.255.0`
  - *NIC0* subnet: `192.200.168.0`
  - *NIC1* subnet: `10.250.2.0`
  - `cgtp0` subnet: `192.200.175.0`

  The floating address triplet is as follows:

  - *NIC0* floating address: `192.200.168.1`
  - *NIC1* floating address: `10.250.2.1`
  - `cgtp0` floating address: `192.200.175.1`

  This address is calculated similarly if you have IP addresses of any other class.

`CMM.MasterLoss.Detection`
: Determine if the absense of a master node in the cluster must be detected by diskless and dataless nodes. Values are `True` and `False`. The default is `False`.

  If `CMM.MasterLoss.Detection` is set to `False`, the diskless and dataless nodes are rebooted when they detect that there is no master node in the cluster for a period of more then 4–5 minutes. If `CMM.MasterLoss.Detection` is set to `True`, the diskless and dataless nodes do not reboot if there is no master node in the cluster. However, the diskless and dataless nodes cannot access directories that would be exported by the master node if such a node existed. You must ensure that diskless and dataless nodes can adapt to this situation.

`Node.Domainid`
: Specify the *domainid* of the cluster. You must modify this parameter. There is no default value.

```
Node.Domainid=250
```

`Node.NodeId`
This parameter specifies the node ID of the current node. There is no default value.

`Node.NIC0`
The name of the first network interface, *NIC0*, used for CGTP. The default value is `hme0`.

```
Node.NIC0=hme0
```

This parameter could be a logical network interface, for example, `hme0:2`.

If you have not installed the CGTP patches and packages and you want to configure a single network link for your cluster, configure this parameter.

`Node.NIC1`
The name of the second network interface, *NIC1*, used for CGTP. The default value is `hme1`.

```
Node.NIC1=hme1
```

This parameter could be a logical network interface, for example, `hme1:2`.

If you have not installed the CGTP patches and packages and you want to configure a single network link for your cluster, do not configure this parameter.

`Node.NICCGTP`
The virtual interface used by CGTP. The default value is `cgtp0`.

```
Node.NICCGTP=cgtp0
```

If you have not installed the CGTP patches and packages and you want to configure a single network link for your cluster, do not configure this parameter.

`Node.UseCGTP`
Specify whether the CGTP is to be used or not. Values are `True` or `False`. The default value is `True`.

```
Node.UseCGTP=True
```

If you have not installed the CGTP patches and packages and you want to configure a single network link for your cluster, set this parameter to `False`.

`Node.Type`
Specify the type of node. Defining this parameter is mandatory. When you create the `nhfs.conf` file on each peer node, specify the type of node in this parameter. The `Node.Type` parameter can have one of the following values:

- `Diskfull`—A master-eligible node
- `Dataless`—A dataless node
- `Diskless`—A diskless node

There is no default value. For more information on types of nodes, see the *Netra High Availability Suite Foundation Services 2.1 6/03 Overview*.

`Node.RNFS.Installed`
Specify whether Reliable NFS is installed on a node. Values are `True` or `False`. The default value is `False`.

```
Node.RNFS.Installed=True
```

Both master-eligible nodes must have the same value for this parameter.

DIRECT LINK PARAMETERS

To prevent a split brain situation, you can connect the serial ports of the master-eligible nodes and configure the following parameters. Split brain is a situation where the network fails and two master nodes are elected in the same cluster. Configure the following parameters if you have connected the serial ports of the master-eligible nodes.

`Node.Direct-Link.Serial.Device`
Specify the serial device, that is, the system's serial ports. There is no default value.

```
Node.Direct-Link.serial.Device=/dev/term/b
```

`Node.Direct-Link.Serial.Speed`
Specify the serial line speed. Valid values are 38400, 57600, 76800, or 115200. The higher the value, the better the link.

There is no default value.

```
Node.Direct-Link.serial.Speed=115200
```

`Cluster.Direct-Link.Backend`
This parameter enables the direct link. If this parameter is not present in the `nhfs.conf` file, the direct link will not be used even if you have connected the serial ports of the master-eligible nodes.

The only value accepted by this parameter is `serial`:

```
Cluster.Direct-Link.Backend=serial
```

`Cluster.Direct-Link.Heartbeat`

Specify the number of seconds between two checks to detect a failure. Therefore, if one master-eligible node receives no heartbeat during the specified period of seconds, the node is alerted that there may be a problem.

There is no default value. For example, specify an interval of 20 seconds as follows:

```
Cluster.Direct-Link.Heartbeat=20
```

CMM PARAMETERS

`CMM.IsEligible`
Specify whether the node is master eligible. Values are `True` or `False`. The default value is `False`.

```
CMM.IsEligible=True
```

`CMM.LocalConfig.Dir`
Specify the directory where the configuration file, `cluster_nodes_table`, is located on each master-eligible node. There is no default value.

```
CMM.LocalConfig.Dir=/etc/opt/SUNWcgha
```

`CMM.Miniconfig.Dir`
Specify the directory where the configuration file, `target.conf`, is located on each master-eligible node. There is no default value.

```
CMM.Miniconfig.Dir=/etc/opt/SUNWcgha
```

`CMM.Miniconfig.File`
Specify the name of the local configuration file, `target.conf`. There is no default value.

```
CMM.Miniconfig.File=target.conf
```

For more information on the CMM parameters, see the `nhcmmd`(1M) man page.

RELIABLE NFS PARAMETERS

`RNFS.StatdAlternatePath`
Specify the alternate `statd` repository. For information about `statd`, see the `statd`(1M) man page.

You must modify this parameter. There is no default value.

```
RNFS.StatdAlternatePath=directory-path
```

where *directory-path* is the path to the `statd` directory. This directory must be on a replicated disk partition.

```
RNFS.StatdAlternatePath=/SUNWcgha/local
```

`RNFS.Slice`
Define the disk partitions that must be replicated by Reliable NFS. You must modify this parameter. The format for this parameter is:

```
RNFS.Slice.x=local-disk-partition  local-bitmap-partition  \
remote-disk-partition  remote-bitmap-partition  mount-flag
```

- *x* is an integer (0, 1, 2) that distinguishes each partition to be replicated.
- *local-disk-partition* and *remote-disk-partition* are the paths to the raw disk partitions on the local and remote nodes.
- *local-bitmap-partition* and *remote-bitmap-partition* are the paths to the scoreboard bitmaps associated to *local-disk-partition* and *remote-disk-partition* respectively.
- *mount-flag* is 0, 1 or 2:

  0 Do not mount the partition.

  1 Mandatory mount. If the partition is not mountable, stop with error.

  2 Best effort mount. Try to mount the partition. If this is not possible, log an error and continue.

There is no default value.

```
RNFS.Slice.0=/dev/rdsk/c0t0d0s3 /dev/rdsk/c0t0d0s5 \
/dev/rdsk/c0t0d0s3 /dev/rdsk/c0t0d0s5 1
```

`RNFS.Share`

Describes the file systems to be shared. You must modify this parameter.

- This parameter uses the same syntax as the `share` command. For more information, see the `share`(1M) man page.
- Each share command must be on its own line. The first string of an `RNFS.Share` line must be either `share` or `/usr/sbin/share`.
- Each `RNFS.Share` line must contain one share command line per partition to be exported by NFS.
- Each `RNFS.Share.`*x* represents a partition to be shared and *x* is an integer (0, 1, 2) that distinguishes each partition to be replicated.

To ensure that the CMM behaves correctly, you must:

- Grant superuser access for all the master-eligible nodes on the exported file system where the `cluster_nodes_table` file resides.
- Use the addresses for the CGTP interface for the master-eligible nodes. If CGTP is not installed, use the *NIC0* addresses.

There is no default value. In the following example, `cgtp6` and `cgtp7` represent the static CGTP addresses of the master-eligible nodes.

```
RNFS.Share.0=share -F nfs -o rw,root=cgtp6:cgtp7 \
-d "SUNWcgha" /SUNWcgha/local/export

RNFS.Share.1=share -F nfs -o rw -d "diskless1" -o \
rw,root=diskless1:diskless1-b:cgtp-diskless1 \
/export/root/netraDISKLESS1
```

If you have diskless nodes in the cluster, add the `cgtp0` address of the diskless nodes.

```
RNFS.Share.0=share -F nfs \
-o rw,root=cgtp6:cgtp7:cgtp8 \
-d "SUNWcgha" /SUNWcgha/local/export
```

`RNFS.CheckReplicatedSlices`

Check the status of replicated slices by continuously scanning them. This property is disabled by default. To enable this property, set `RNFS.CheckReplicatedSlices` to True:

```
RNFS.CheckReplicatedSlices=True
```

This check is only performed when the cluster is in the `synchro:READY` state. To determine the synchronization state of the cluster, run the `nhcmmstat` command. If you enable this property and the cluster is not in the `synchro:READY` state the following action occurs:

- An error message is displayed on the master node.
- The cluster is put in the `synchro:NEEDED` state to prevent a switchover occurring. If a switchover occurs, the master might not have access to the most up to date data.

`RNFS.CheckReplicatedSlicesInterval`
Set the time between two successive reads of the replicated slices. Values are a number of milliseconds. The default value is 10 milliseconds. The `RNFS.CheckReplicatedSlicesInterval` property is ignored if the `RNFS.CheckReplicatedSlices` property is not set to `True`.

```
RNFS.CheckReplicatedSlicesInterval=time in milliseconds
```

`RNFS.EnableSync`
Determine when disk synchronization occurs. Values are `True` and `False`. The default is `True`.

If `RNFS.EnableSync` is set to `True`, disk synchronization is triggered at startup.

If `RNFS.EnableSync` is set to `False`, you delay the start of disk synchronization until you use the `nhenablesync` command. For more information on this command, see the `nhenablesync`(1M) man page.

```
RNFS.EnableSync=False
```

`RNFS.SyncType`
Specify the method used for synchronization between slices on master-eligible nodes. Values are `FS` or `RAW`. The default value is `FS`.

If `RNFS.SyncType` is set to `FS`, this feature is enabled. The time taken for a full slice synchronization is reduced because only the blocks used by the slice's file-system are replicated. If `RNFS.SyncType` is set to `RAW`, this property is disabled.

Both master-eligible nodes must have the same value for the `RNFS.SyncType` parameter. If you change the value of the `RNFS.SyncType` property, reboot the master-eligible nodes one at a time. For information on rebooting cluster nodes, see "Shutting Down and Restarting a Cluster" in the Netra High Availability Suite Foundation Services 2.1 6/03 Cluster Administration Guide. The `RNFS.SyncType` property is only valid for master-eligible nodes.

`RNFS.SerializeSync`
Determine how disk synchronization is performed. Values are `TRUE` and `FALSE`. The default value is `FALSE`.

Synchronization is necessary following a switchover or the vice-master booting, or when you request a full replication. In these circumstances, if `RNFS.SerializeSync` is set to `FALSE`, Reliable NFS starts the synchonization of all slices at the same time. If `RNFS.SerializeSync` is set to `TRUE`, slices are synchronized one slice at a time. This reduces the network and disk overhead but increases the time it takes for the vice-master to synchronize with the master. During this time, the vice-master is not eligible to take on the role of master.

For more information on the Reliable NFS parameters, see the `nhcrfsd`(1M) man page.

Note – Some parameters in this file require a single value while others require multiple values.

```
RNFS.StatdAlternatePath=/SUNWcgha/local

RNFS.Slice.0=/dev/rdsk/c0t0d0s3 /dev/rdsk/c0t0d0s5 /dev/rdsk/c0t0d0s3 \
/dev/rdsk/c0t0d0s5 1

RNFS.Slice.1=/dev/rdsk/c0t0d0s4 /dev/rdsk/c0t0d0s6 /dev/rdsk/c0t0d0s4 \
/dev/rdsk/c0t0d0s6 1

RNFS.Share.0=share -F nfs -o rw,root=b14-ct400-10-cgtp:b14-ct400-20-cgtp \
-d "SUNWcgha" /SUNWcgha/local/export

RNFS.Share.1=share -F nfs -o rw -d "Export" /export
```

NODE STATE MANAGER PARAMETERS

Configure these parameters to specify the external floating address assigned to the master node. The following parameters must be configured when you use the Node State Manager.

`NSM.Exec.MasterDir`
The directory containing the scripts for the master node. There is no default value.

```
NSM.Exec.MasterDir=/opt/SUNWcgha/actions/master
```

`NSM.Exec.ViceMasterDir`
The directory containing the scripts for the vice-master node. There is no default value.

```
NSM.Exec.ViceMasterDir=/opt/SUNWcgha/actions/vicemaster
```

`NSM.Log.MasterDir`
The log file directory for the master node. There is no default value.

```
NSM.Log.MasterDir=/var/run/SUNWcgha/actions/master
```

`NSM.Log.ViceMasterDir`
The log file directory for the vice-master node. There is no default value.

```
NSM.Log.ViceMasterDir=/var/run/SUNWcgha/actions/vicemaster
```

`NSM.External.Master.Address`
The external floating address that is always assigned to the master node.

`NSM.External.Master.Address=`*IP-address/netmask-size*

The *netmask-size* value is optional. There is no default value.

```
NSM.External.Master.Address=192.168.12.39
```

`NSM.External.Master.Broadcast`
The broadcast address. This parameter is optional. If not configured, the broadcast address is automatically determined based on the IP address and netmask. There is no default value.

```
NSM.External.Master.Broadcast=192.168.255.255
```

`NSM.External.Master.NIC`
The physical or logical network interface for the external floating address of the master node. There is no default value.

```
NSM.External.Master.NIC=hme0:3
```

`NSM.External.Master.RouteAdd`
Set the arguments to be passed to the `route add` command when a master-eligible node becomes a master node. This parameter is optional. If not defined, no routes are defined. There is no default value.

```
NSM.External.Master.RouteAdd=default 192.168.12.250
```

The preceding example generates the following command:

```
route add default 192.168.12.250
```

`NSM.External.Master.RouteDelete`
Set the arguments to be passed to the `route delete` command when a node loses the master role. This parameter is optional. There is no default value.

```
NSM.External.Master.RouteDelete=default 192.168.12.250
```

The preceding example generates the following command:

```
route delete default 192.168.12.250
```

WATCHDOG TIMER PARAMETERS

The following parameters are optional and must be configured only if you have installed the Watchdog Timer on the nodes of your cluster.

`WATCHDOG.NhasWatchdog`
If this parameter is set to `true`, the product enables the watchdog and initiates the patting of the watchdog.

Options are `true` or `false`. The default is `false`.

```
WATCHDOG.NhasWatchdog=false
```

`WATCHDOG.ShutDownTimeout`
The maximum number of milliseconds to wait for the system to reach `init` run level 6 during a shutdown. See the `init`(1M) man page.

The value must be provided in milliseconds. The default value is `5000`.

```
WATCHDOG.ShutDownTimeout=6000
```

`WATCHDOG.OsTimeout`
The maximum number of milliseconds to wait between the system boot at `init` run level 2 and `init` run level 6. See the `init`(1M) man page.

The value must be provided in milliseconds. The default value is `3000`.

```
WATCHDOG.OsTimeout=5000
```

`WATCHDOG.PattingIntvl`
The interval after which the hardware watchdog is re-enabled by the Watchdog Timer daemon, `nhwdtd`. The value must be provided in milliseconds and must be at least 1000 milliseconds less than the values of the `WATCHDOG.OsTimeout` and `WATCHDOG.ShutDownTimeout` parameters. The default value is `2000`.

```
   patting_intvl=5000
```

EXAMPLES

EXAMPLE 1 Example of the `nhfs.conf` File on a Master-Eligible Node

```
### Common part
Node.DomainId=69
Node.NIC0=hme0:1
Node.NIC1=hme1
Node.NICCGTP=cgtp0
Node.UseCGTP=True
Node.type=Diskfull

### Cluster part
Cluster.Master.Id=1

### CMM part
CMM.IsEligible=True
CMM.LocalConfig.Dir=/etc/opt/SUNWcgha

### RNFS part
RNFS.StatdAlternatePath=/SUNWcgha/local

RNFS.Slice.0=/dev/rdsk/c0t0d0s3 /dev/rdsk/c0t0d0s5 /dev/rdsk/c0t0d0s3 \
/dev/rdsk/c0t0d0s5 1
RNFS.Slice.1=/dev/rdsk/c0t0d0s4 /dev/rdsk/c0t0d0s6 /dev/rdsk/c0t0d0s4 \
/dev/rdsk/c0t0d0s6 1

RNFS.Share.0=share -F nfs -o rw,root=b14-ct400-10-cgtp:b14-ct400-20-cgtp \
-d "SUNWcgha" /SUNWcgha/local/export
RNFS.Share.1=share -F nfs -o rw -d "Export" /export

RNFS.Installed=True

### NSM Part
NSM.Exec.MasterDir=/opt/SUNWcgha/actions/master
NSM.Exec.ViceMasterDir=/opt/SUNWcgha/actions/vicemaster

NSM.Log.MasterDir=/var/run/SUNWcgha/actions/master
NSM.Log.ViceMasterDir=/var/run/SUNWcgha/actions/vicemaster

NSM.External.Master.Address=192.168.12.39
NSM.External.Master.NIC=hme0:3
NSM.External.Master.Broadcast=192.168.255.255

NSM.External.Master.RouteAdd=default 192.168.12.250
NSM.External.Master.RouteDelete=default 192.168.12.250
```

EXAMPLE 2 Example of the `nhfs.conf` File on a Diskless Node

```
### Common part
Node.DomainId=250
Node.NIC0=hme0:1
Node.NIC1=hme1
Node.NICCGTP=cgtp0
Node.UseCGTP=True
Node.type=Diskless
```

EXAMPLE 2 Example of the `nhfs.conf` File on a Diskless Node *(Continued)*

```
### CMM part
CMM.IsEligible=False
CMM.LocalConfig.Dir=/etc/opt/SUNWcgha
CMM.CurrentNodeName=netraDISKLESS1
```

EXAMPLE 3 Example of the `nhfs.conf` File on a Dataless Node

```
Node.NICCGTP=cgtp0
Node.UseCGTP=True
Node.NIC1=eri1
Node.NIC0=eri0
Node.DomainId=250
Node.Type=Dataless

CMM.IsEligible=False
CMM.LocalConfig.Dir=/etc/opt/SUNWcgha
CMM.CurrentNodeName=netraDATALESS1
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcdt
Interface Stability	Evolving

SEE ALSO

`init`(1M), `nhcrfsd`(1M), `nhcmmd`(1M), `share`(1M), `nhenablesync`(1M), `nhpmd`(1M), `nhnsmd`(1M), `cluster_nodes_table`(4)

NAME

nhpmd.conf – enable you to override the number of times the PMD retries a daemon it monitors

SYNOPSIS

/etc/opt/SUNWcgha/nhpmd.conf

DESCRIPTION

The `nhpmd.conf` file is optional file. If you create this file, you can specify the number of times the PMD tries to restart a failed daemon (*retry-count*) by providing the number as an option to the `nhpmdadm` command.

The *retry-count* values you specify in the `nhpmd.conf` file supersede the values present in the HA scripts. If the `nhpmd.conf` file does not exist or if there is no entry for a particular daemon, the number of retries specified in the HA scripts is used.

The line format in `nhpmd.conf` is:

```
<nametag>_RetryCount = <retry_count_value>
```

For example for the `statd` daemon the entry would be:

```
nfs.client.statd.crfs_RetryCount = 2
```

Note – You must insert spaces or tabs before and after "=".

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcma
Interface Stability	Evolving

SEE ALSO

`nhpmd`(1M), `nhpmdadm`(1M), `nhpmdadmwrapper`(1M)

NAME

nma.notifs.txt – Node Management Agent (NMA) configuration file for SNMP trap notifications.

SYNOPSIS

`nma.notifs.txt`

DESCRIPTION

The *installDir*`/etc/opt/SUNWcgha/nma.notifs.txt` file defines the NMA SNMP trap notification types and the target to which each type will be sent. By default this file is located at `/etc/opt/SUNWcgha/nma.notifs.txt`. Replace *installDir* with the root installation directory if the root installation directory is not `/`.

EXTENDED DESCRIPTION

An entry in `nma.notifs.txt` takes the form
`notificationEntry=`*notifIdentifier*`,`*tag*`,`*NotifType*`,`*persistency*

notifIdentifier	Unique notification ID.
tag	Tag of the target.
NotifType	Choose `1` for `trap`. The NMA does not send notifications of any other type.
persistency	Storage type. Choose `3` for non-volatile storage.

EXAMPLES

The following example is the default file `nma.notifs.txt`.

EXAMPLE 1 Example `nma.notifs.txt` File

```
#########################################################
# Please refer to the [RFC2573] for information details
#########################################################
# notificationEntry = <notifIdentifier>, <tag>,\
# <NotifType>, <persistency>
# where :
#    <notifIdentifier> = Unique ID of the notification
#    <tag>             = Tag to use to select targets
#                              (refer to targets.txt <tagList>)
#    <notifType>       = (1) Trap, (2) Inform
#    <persistency>     = Persistency Type to use for the row

# Trap notification must be sent to targets containing tag in taglist.
notificationEntry=notif1,trap,1,3
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhmas
Interface Stability	Evolving

SEE ALSO

`nma`(1M), `nma.params.txt`(4), `nma.properties`(4), `nma.security`(4), `nma.targets.txt`(4)

NAME

nma.params.txt – file to configure the Node Management Agent (NMA) SNMP parameters.

SYNOPSIS

`nma.params.txt`

DESCRIPTION

The *installDir*`/etc/opt/SUNWcgha/nma.params.txt` file defines the communication, implementation and accessibility configuration of SNMP in the NMA. By default this file is located at `/etc/opt/SUNWcgha/nma.params.txt`. Replace *installDir* with the root installation directory if the root installation directory is not `/`.

EXTENDED DESCRIPTION

An entry in the `nma.params.txt` uses the following syntax:

`paramsEntry=`*paramsIdentifier*`,`*processingModel*`,`*securityModel*`,`*securityName*`,\`
securityLevel`,`*persistency*

paramsIdentifier	ID of the SNMP parameter set.
processingModel	Values can be `0`, `1` or `3` for SNMPv1, SNMPv2 or SNMPv3 repectively.
securityModel	`0`, `1` or `3` for SNMPv1, SNMPv2 or SNMPv3 repectively.
securityName	The name of the principal to use.
securityLevel	`1` for no authentication and no cyphering, `2` for authentification with no cyphering, or `3` for authentification and cyphering.
persistency	Storage type. Choose `3` for non-volatile storage.

EXAMPLES

The following example is the default `nma.params.txt` file.

EXAMPLE 1 Example `nma.params.txt` File

```
#########################################################
# Please refer to the [RFC2573] for information details
#########################################################
# paramsEntry=<paramsIdentifier>,<processingModel>,\
# <securityModel>,<securityLevel>,\
# <securityName>,<securityLevel>,<persistency>
# where :
#  <paramsIdentifier> = Unique ID of the SNMP parameters
#  <processingModel>  = (0) SNMP V1, (1) SNMP V2c, (3) SNMP V3
#  <securityModel>    = (0) SNMP V1, (1) SNMP V2c, (3) SNMP V3
#  <securityName>     = Principal to use
#  <securityLevel>    = (1) NoAuthNoPriv, (2) AuthNoPriv, (3) AuthPriv
#  <persistency>      = Persistency type of the row

# SNMP parameters V2
paramsEntry=snmpV2,1,1,public,1,3
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhmas
Interface Stability	Evolving

SEE ALSO

`nma`(1M), `nma.notifs.txt`(4), `nma.properties`(4), `nma.security`(4), `nma.targets.txt`(4)

NAME

nma.properties – Node Management Agent (NMA) configuration file.

SYNOPSIS

/etc/opt/SUNWcgha/nma.properties

DESCRIPTION

The file .

The *installDir*/etc/opt/SUNWcgha/nma.properties file defines the communication, implementation, and accessibility configuration of the NMA. By default this file is located at /etc/opt/SUNWcgha/nma.properties. Replace *installDir* with the root installation directory if the root installation directory is not /.

The nma.properties file contains configuration options for the NMA, RFC 2573, and Java Dynamic Management Kit (Java DMK) and Java Managemetn eXtensions (JMX).

This file is in ASCII format. Comment lines begin with the comment mark (#). Properties consist of a keyword followed by an equals (=) sign followed by the parameter value, of the form:

Keyword=Value

NMA Properties

The following properties enable you to configure the NMA.

com.sun.nhas.ma.adaptors.html.enabled
Enable the HTML adaptor. Options are true or false. By default, the value is true. For example:

```
com.sun.nhas.ma.adaptors.html.enabled=true
```

com.sun.nhas.ma.adaptors.html.port
Define the HTML adaptor port. By default, the port is 8082. For example:

```
com.sun.nhas.ma.adaptors.html.port=8082
```

com.sun.nhas.ma.adaptors.snmp.enabled
Enable the SNMP adaptor. Options are true or false. By default, the value is true. For example:

```
com.sun.nhas.ma.adaptors.snmp.enabled=true
```

com.sun.nhas.ma.adaptors.snmp.port
Define the SNMP adaptor port number. By default, this value is 8085. For example:

```
com.sun.nhas.ma.adaptors.snmp.port=8085
```

com.sun.nhas.ma.adaptors.snmp.trap.port
Define the UDP port to which the SNMP agent sends traps if the adaptors.snmp.rfc2573.enabled property is set to false. By deault, the value is 8086. For example:

```
com.sun.nhas.ma.adaptors.snmp.trap.port=8086
```

com.sun.nhas.ma.connectors.rmi.enabled
Enable the RMI connector. Options are true or false. By default, the value is true. For example:

```
com.sun.nhas.ma.connectors.rmi.enabled=true
```

`com.sun.nhas.ma.connectors.rmi.port`
Define the RMI connector port. By default, the port is `1098`. For example:

```
com.sun.nhas.ma.connectors.rmi.port=8082
```

`com.sun.nhas.ma.connectors.http.enabled`
Enable the HTTP connector. Options are `true` or `false`. By default, the value is `true`. For example:

```
com.sun.nhas.ma.connectors.http.enabled=true
```

`com.sun.nhas.ma.connectors.http.port`
Define the HTTP connector port. By default, the port is `8081`. For example:

```
com.sun.nhas.ma.connectors.http.port=8081
```

`com.sun.nhas.ma.connectors.http.client`
Define the maximum number of HTTP connector clients. By default, the value is `9999`. For example:

```
com.sun.nhas.ma.connectors.http.client=9999
```

`com.sun.nhas.ma.nhas.configuration.path`
Define the path to the `nhfs.conf` file. By default, the path is `/etc/opt/SUNWcgha/nhfs.conf`. For example:

```
com.sun.nhas.ma.nhas.configuration.path=/etc/opt/SUNWcgha/nhfs.conf
```

`com.sun.nhas.ma.cgtp.polling`
Define the period of CGTP information refresh, expressed in milliseconds. By default, the value is `20000`. The minimum value is `5000`. Set this property to `-1` to disable polling. For example:

```
com.sun.nhas.ma.cgtp.polling=20000
```

`com.sun.nhas.ma.pmd.polling`
Define the period of Daemon Monitor information refresh, expressed in milliseconds. Set to `-1` to disable polling. By default, the value is `20000`. For example:

```
com.sun.nhas.ma.pmd.polling=20000
```

`com.sun.nhas.ma.pmd.cache.validity`
Define the time for which cached Daemon Monitor data is valid, expressed in milliseconds. Set to `-1` to disable the use of the cache. By default the value is `2000`. For example:

```
com.sun.nhas.ma.pmd.cache.validity=2000
```

`com.sun.nhas.ma.operation.flag`
Enable the access to the MBeans operations provided by the NMA. Options are `true` or `false`. By default, the value is `true`. For example:

```
com.sun.nhas.ma.operation.flag=true
```

`com.sun.nhas.ma.cascading.enabled`
Enable cascading. Options are `true` or `false`. To disable cascading, By default, this property is disabled and cascading is enabled. To disable cascading, uncomment this property. For example:

```
com.sun.nhas.ma.cascading.enabled=false
```

`com.sun.nhas.ma.cascading.retries.max`
Define the maximum number of retries to be made before aborting the attempt to activate a cascading agent. By default, the value is `50`. For example:

```
com.sun.nhas.ma.cascading.retries.max=50
```

`com.sun.nhas.ma.cascading.retries.delay`
Define the delay period between each retry of the attempted activation of a cascading agent, expressed in milliseconds. By default, the value is `2000`. For example:

```
com.sun.nhas.ma.cascading.retries.delay=2000
```

`com.sun.nhas.ma.cascading.comm.protocol`
Define the communication protocol to be used when cascading to sub-agents. By default, the value is `rmi`. For example:

```
com.sun.nhas.ma.cascading.comm.protocol=rmi
```

`java.rmi.server.hostname`
When using RMI to connect to the agent, set this property to the hostname or IP address that the manager uses to communicate to the agent. RMI is normally reserved for cascading, and the default value is the CGTP address of the node. Note that changing this property may affect cascading. By default, this property is disabled and there is no default value. Uncomment this line and provide an IP address to set this property. For example:

```
java.rmi.server.hostname=10.250.1.10
```

RFC 2573 Properties

The following properties control the NMA implementation of Internet Engineering Task Force (IETF) RFC 2573. Details of RFC 2573 are accessible at `http://www.ietf.org/rfc/rfc2573.txt`

`adaptors.snmp.rfc2573.enabled`
Enable `RFC2573` support. Options are `true` or `false`. By default, the value is `true`. For example:

```
adaptors.snmp.rfc2573.enabled=true
```

`adaptors.snmp.rfc2573.v1v2set.enabled`
Enable SNMPv1 or SNMPv3 requests to be made to the MIB. Options are `true` or `false`. By default, the value is `false`. For example:

```
adaptors.snmp.rfc2573.v1v2set.enabled=false
```

`adaptors.snmp.rfc2573.target.addr.file`
Define the path to the `nma.targets.txt` file. By default, the value is `/etc/opt/SUNWcgha/nma.targets.txt`. For example:

```
adaptors.snmp.rfc2573.target.addr.file=/etc/opt/SUNWcgha/nma.targets.txt
```

See `nma.targets.txt`(4) for more information.

`adaptors.snmp.rfc2573.target.params.file`
Define the location of the `nma.params.txt` file. By default, the value is `/etc/opt/SUNWcgha/nma.params.txt`.

See `nma.params.txt`(4) for more information.

`adaptors.snmp.rfc2573.notification.file`
Define the location of the `nma.notifs.txt` file. By default, the value is `/etc/opt/SUNWcgha/nma.notifs.txt`. For example:

```
adaptors.snmp.rfc2573.notification.file=/etc/opt/SUNWcgha/nma.notifs.txt
```

See `nma.notifs.txt`(4) for more information.

Java DMK and JMX Properties

The following properties define the Java DMK and JMX configuration file locations.

`jdmk.security.file`
Define the location of the file that defines the SNMPv3 security parameters. This path is local to each node. By default, the value is `/etc/opt/SUNWcgha/nma.security`. For example:

```
jdmk.security.file=/etc/opt/SUNWcgha/nma.security
```

See `nma.security`(4) for more information.

`jdmk.uacl.file`
Define a specific user-based ACL file for SNMPv3 requests. A template is available at `/etc/opt/SUNWcgha/nma.uacl.template`. By default, this property is disabled. To define a user-based ACL file, uncomment this property and provide the path to the file. For example:

```
jdmk.uacl.file=/etc/opt/SUNWcgha/nma.uacl
```

`jdmk.acl.file`
Define a specific ACL file for trap destination and access control configuration. A template is available at `/etc/opt/SUNWcgha/nma.acl.template`. By default, this property is disabled. To define an ACL file, uncomment this property and provide the path to the file. For example:

```
jdmk.acl.file=/etc/opt/SUNWcgha/nma.acl
```

`jmx.serial.form`
Enable compatibility with clients using JMX 1.0 serialization. This property is required when using an agent or client based on Java DMK versions older than 5.0. By default, this property is disabled. To enable compatibility, uncomment this property and proviude a serial number. For example:

```
jmx.serial.form=1.0
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhmas
Interface Stability	Evolving

SEE ALSO

`nma`(1M), `nma.notifs.txt`(4), `nma.params.txt`(4), `nma.security`(4), `nma.targets.txt`(4)

NAME

nma.security – Node Management Agent (NMA) configuration file for SNMP security.

SYNOPSIS

`nma.security`

DESCRIPTION

Configure the *installDir*`/etc/opt/SUNWcgha/nma.security` file to define user entries which are used to identify authorised users of the NMA SNMP Management Information Bases (MIBs). By default this file is located at `/etc/opt/SUNWcgha/nma.security`. Replace *installDir* with the root installation directory if the root installation directory is not `/`.

EXTENDED DESCRIPTION

Modify variables in the `nma.security` file as follows:

`localEngineID` The engine ID used to configure the SNMP engine. This property is optional. If you do not provide an engine ID, the engine ID will be generated by the system on the basis of the CGTP address of the node.

`userEntry` A comma-seperated list of the format:

`userEntry=`*engine-ID*`,`*user-name*`,`*security-name*`,`*authentication-algorithm*`,`*authentication-key*

engine-ID and the *user-name* are mandatory parameters. All the other parameters are optional.

engine-ID The SNMP engine ID.

user-name The unique ID of this user.

security-name The principal on whose behalf SNMP messages will be generated.

authentication-algorithm The type of authentication algorithm to be used for this user. The following algorithms are permitted:

- `usmHMACMD5AuthProtocol`
- `usmHMACSHAAuthProtocol`
- `usmNoAuthProtocol`

authentication-key The user password

EXAMPLES

The following example is the default `nma.security` file:

EXAMPLE 1 Example `nma.security` file

```
# Each SNMP V3 engine has its own security file.
# You have one file on the Node management side
# and one on the manager side.
# Both files will have a very similar configuration.
# If you don't provide an engine id and or an engine boots,
# they are computed by the NMA engine based on the
# CGTP address of the node and the SNMP port configured
# in the nma.properties.

# localEngineID=<your engine id>
```

EXAMPLE 1 Example `nma.security` file *(Continued)*

```
userEntry=localEngineID,defaultUser,null,usmHMACMD5AuthProtocol,mypasswd
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhmas
Interface Stability	Evolving

SEE ALSO

`nma`(1M), `nma.notifs.txt`(4), `nma.params.txt`(4), `nma.properties`(4), `nma.targets.txt`(4)

NAME

nma.targets.txt – configuration file for the Node Management Agent (NMA) SNMP trap targets.

SYNOPSIS

`nma.targets.txt`

DESCRIPTION

The *installDir*`/etc/opt/SUNWcgha/nma.targets.txt` file defines the targets to which the NMA sends SNMP traps. By default this file is located at `/etc/opt/SUNWcgha/nma.targets.txt`. Replace *installDir* with the root installation directory if the root installation directory is not `/`.

EXTENDED DESCRIPTION

An entry in the `nma.targets.txt` file uses the syntax:

`targetsEntry=`*targetIdentifier* , *domain* , *host/port* , *timeout* , *retry* , \
tagList , *paramsIdentifier* , *persistency*

targetIdentifier	Unique ID of the target
domain	The OID of the UDP or TCP domain
host/port	The address of the target in the format: *hostname-or-IP-address/port-number* If you are using a UDP domain, then the format is: *hostname-or-IP-address"/"port number*
timeout	Timeout after which requests are resent
retry	The number of request send retries
tagList	The list of tags that identify the targets
paramsIdentifier	ID of the SNMP parameter set to be used for communication
persistency	Storage type. Choose `3` for non-volatile storage.

EXAMPLES

The follow example is the default `nma.targets.txt` file.

EXAMPLE 1 `nma.targets.txt`

```
#########################################################
# Please Refer to the [RFC2573] for information details
#########################################################
#
# targetsEntry=<targetIdentifier>,<domain>,\
# <host/port>,<timeout>,<retry>,\
# <tagList>,\
# <paramsIdentifier>,<persistency>
#    where :
#       <targetIdentifier> = Unique ID of the targets
#       <domain>           = Could be UDP or TCP (OID of the domain)
#       <host/port>        = Address of the target. If UDP domain,
#                  then the format is host"/"port
#       <timeout>          = Timeout after which requests are sent again
#       <retry>            = retry count number
```

EXAMPLE 1 `nma.targets.txt` *(Continued)*

```
#       <tagList>          = list of tags to allow the selection of the
#                    target
#       <paramsIdentifier> = Identifier of the SNMP parameter set to
#                    use to communicate with
#       <persistency>      = persistency type
#
# Target "localhost" wants to receive traps on port 8086 using the SNMP
# parameters V2
targetsEntry=managerV2,snmpUDPDomain,localhost/8086,10000,2,trap,snmpV2,3
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhxxx
Interface Stability	Evolving

SEE ALSO

`nma.security`(4)`nma`(1M)

NAME

nodeprof.conf – `permit the customization of Solaris installation`

SYNOPSIS

`nodeprof.conf`

DESCRIPTION

When installing a cluster with `nhinstall`, you can customize the Solaris installation using the `nodeprof.conf` file. When the `nodeprof.conf` file is present in your configuration directory, its contents supersede the default profile used by `nhinstall`.

The templates for the configuration files are contained in the `/opt/SUNWcgha/config.standard` directory with `.template` extensions. Copy the configuration files to a local directory on the installation server as follows:

```
# mkdir config-file-directory
# export NHOME=/opt/SUNWcgha/config.standard
# cd config-file-directory
# cp $NHOME/nodeprof.conf.template nodeprof.conf
```

Note – All the configuration files must be in the same local directory on the installation server.

For information on the format of the `nodeprof.conf` file see "Preparing Custom JumpStart Installations (Tasks)" in the *Solaris Installation Guide*. Do not define disks and partitions in the `nodeprof.conf` file because `nhinstall` automatically adds these configuration details to the file. This means that you must not define the `boot_device`, `root_device`, `filesys` or `usedisk` commands because the configuration information for the disks is defined in the `cluster_definition.conf` file.

The metacluster which is defined the first time you use the `cluster` command in the `nodeprof.conf` file is used as the metacluster for diskless nodes and is given as an argument to the `smosservice` command.

If you do not want to install the same Solaris distribution on diskless nodes as you have installed on master-eligible nodes, create a `diskless_nodeprof.conf` file. For more information, see the `diskless_nodeprof`(4) man page.

Note – You cannot use `nhinstall` to install software on dataless nodes.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhins

SEE ALSO

`diskless_nodeprof.conf`(4), `nhinstall`(1M), *Solaris Installation Guide*

NAME

software.conf – SMCT configuration file to define the additional software packages and patches to be deployed to each node group

SYNOPSIS

SMCT_CONFIG_DIR/services/*node_group_name*.*architecture*.*operating_system*

config-dir/**services/***node_group_name*.*architecture*.*operating_system*

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

The software configuration file is an optional file that defines the additional software packages and patches to be deployed to a specific node group. There must be a software configuration file for each node group that contains user applications.

The software configuration file must be named in the following format:

node_group_name.*architecture*.*operating_system*

- *node_group_name*

  Name of the node group, as defined in the `cluster.conf` file, for example, `master_el`.

- *architecture*

  This must be set to `SPARC`.

- *operating_system*

  This must be set to `SOLARIS`.

For example, the software configuration file for a node group called `master_el` would be `master_el.SPARC.SOLARIS`.

PARAMETERS FOR ADDING PACKAGES

Use the following syntax to describe software packages:

```
{ ELEMENT software software_location
              type PACKAGE
              mode mode_value
              [name name_value]
              [version version_value]
              [base base_dir]
              [adminFile adminFile_location]
              [responseFile responsefile_location]
      }*
```

- *software_location*

  Location of the package expressed as a URI.

- *mode_value*

  Package installation mode. This value can be one of the following:

  - `LOCAL`

    The installation directory relative to the node root file system.

  - `SHARED`

The installation base directory relative to the diskless node group shared file system, `/usr`, or the shared file system mount point for the master-eligible nodes.

- *name_value*

  Name of the package. If this parameter is defined, it must match the package name.

- *version_value*

  Version of the package. If this parameter is defined, it must match the package version.

- *base_dir*

  Package installation directory. This parameter overrides the default installation directory.

- *adminFile_location*

  Location of the special package administration file used by the `pkgadd` tool.

- *responsefile_location*

  Location of the special package response file used by the `pkgadd` tool.

PARAMETERS FOR ADDING PATCHES

Use the following syntax to describe the software patches:

```
{ ELEMENT software software-location
              type type-value
              mode mode-value
              [name name-value]
              [version version-value]
      }*
```

- *software_location*

  Location of the patch expressed as a URI.

- *type_value*

  Package type. This value can be one of:

  - `PRE_PATCH`

    Installs the patch after the Solaris operating system packages have been installed, but before the Foundation Services packages and the user-defined applications have been installed.

  - `POST_PATCH`

    Installs the patch after the Solaris operating system packages, the Foundation Services packages, and the user-defined applications have been installed.

- *mode_value*

  The patch installation mode. This value can be one of the following:

  - `LOCAL`

    The installation base directory relative to the root file system of the node.

  - `SHARED`

The installation base directory relative to the shared file system of the diskless node group, `/usr`, or the shared file system mount point for the master-eligible nodes.

- *name_value*

  Name of the patch. If this parameter is defined, it must match the patch name.

- *version_value*

  Version of the patch. if this parameter is defined, it must match the patch version.

PARAMETERS FOR ADDING SOFTWARE DISTRIBUTIONS

Use the following syntax to describe the software distribution:

```
{ ELEMENT software software-location
                [installScript install-script_location
                type type-value
                [name name-value]
                [version version-value]
        }*
```

- *software_location*

  Location of the software distribution expressed as a URI.

- *install_script_location*

  Location of the installation script expressed as a URI.

- *type_value*

  Software distribution type. This parameter must be set to `GENERIC`.

- *name_value*

  Name of the software.

- *version_value*

  Version of the software.

EXAMPLE

The following is an example of a software configuration file `master_el.SPARC.SOLARIS`, for a node group `master_el`:

```
ELEMENT software 'file:/<DS_DIR>/directory-5.1-us.sparc-sun-solaris2.8.tar.gz'
        installScript 'file:/<DS_DIR>/ldap-install.sh'
        type GENERIC name ldap-ds version 5.1

ELEMENT software 'file:/<APPS_PKG_DIR>/SUNWnhhad' mode SHARED
        base /<--SWLID-->/opt
        type PACKAGE

ELEMENT software 'file:/<APPS_PKG_DIR>/SUNWnhccs' mode SHARED
        base /<--SWLID-->/opt
        type PACKAGE

ELEMENT software 'file:/<APPS_PKG_DIR>/SUNWnhmes' mode SHARED
        base /<--SWLID-->/opt
        type PACKAGE

ELEMENT software 'file:/<APPS_PKG_DIR>/SUNWnhapp' mode LOCAL
        responseFile 'file:/export/home/LDAP/SUNWnhapp.resp'
```

```
        type PACKAGE

ELEMENT software 'file:/<SMCT_SOL_DIR>/Solaris_9/Product/SUNWxcu4' mode LOCAL
        type PACKAGE

ELEMENT software 'file:/<SOL_PAT_DIR>/108434-06' mode LOCAL
        type POST_PATCH  version 06

ELEMENT software 'file:/<SOL_PAT_DIR>/108435-06' mode LOCAL
        type POST_PATCH  version 06
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO

`slcreate`(1M)

NAME

target.conf – local configuration file

SYNOPSIS

/etc/opt/SUNWcgha/target.conf

DESCRIPTION

The local configuration file, `target.conf`, contains the cluster ID, attributes, and election roles for each node in the cluster. For example, the `target.conf` file will specify whether the node is a master-eligible node. The `target.conf` file is located on each node on the cluster and contains a description of the node on which it is located.

WARNINGS

Modify this file only when you are manually fixing a problem on the cluster as described in the `cluster_nodes_table`(4) man page.

EXTENDED DESCRIPTION

The following is an example of a `target.conf` file:

```
domain_id: 128          # Cluster ID
attributes: -           # Local nodes attributes
election: 5             # Election round number
role: MASTER            # Role
```

For an explanation of the fields in the example, see the `cluster_nodes_table`(4) man page.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhcma
Interface Stability	Evolving

SEE ALSO

`nhcmmd`(1M), `cluster_nodes_table`(4), `nhadm`(1M), `nhfs.conf`(4)

NAME

userapp.conf – SMCT structured list of user-defined configuration data to be installed at cluster installation and startup

SYNOPSIS

`SMCT_CONFIG_DIR/models/config.`*node_group_name*

*config-dir***`/models/config.`***node_group_name*

DESCRIPTION

Caution – Do not use the SMCT tool with the current patch level of the Foundation Services product.

An SMCT configuration file for user applications is required for each cluster node group that is to be installed with user applications during SMCT configuration stage two. This file is used as a parameter of the `slconfig` command. For more information on `slconfig`, see the `slconfig`(1M) man page.

This configuration file contains a structured list of user-defined configuration data to be installed at specific stages of cluster startup, together with the user applications. User-defined configuration data can be the following:

- User application data files that contain data to be added to the user application.

  For example, if the user application is a database package, the user application data file could be data to populate the database.
- Application installation scripts that add the user application data to the user applications.

  For example, you can specify a script or scripts to populate the database with the application data to create new records and entries.

The configuration file for user applications uses the terms `config` and `file` to describe the running order of the application installation scripts and the user application data that are to be installed.

The configuration file contains a chain for three stages of the cluster installation and startup. The stages are the following:

- At the final stage of the Solaris JumpStart installation of the flash archive.
- At run level 2, the Foundation Services packages are installed but they are not started.
- At run level 3, the Foundation Services are started and running.

You can create a configuration file for each node group, using the following file naming rules:

`config.`*node_group_name*

node_group_name is the name of the node group as defined in the file `cluster.conf`, for example, `soft_switch`. The user application configuration file name for the node group `soft_switch` would be `config.soft_switch`.

When creating the configuration file for user applications, consider the following:

- You cannot have more than two `config` element chains in a file.

- All files in a `config` chain must be of the same type (`PRE_SCRIPT` or `POST_SCRIPT`).
- A `config` element cannot be chained to itself.
- There must not be circular dependencies between `config` elements.
- There must be no infinite `config` chains.

PARAMETERS

This section describes the parameters used in the user application configuration file:

```
{ ELEMENT config config-name
                  [next config-name]
          INVOLVE {file file-name}+
        }+

{ ELEMENT file file-name
                  location file-location
                  type file-type
                  [runLevel run-level-value] }+
        }+
```

- *config_name*

  The name of the `config` element. Each `config` element contains an ordered sequence of user-defined configuration data files to be executed or installed, for example, two user application data files and an application installation script.

- *file_name*

  The unique file name for the user application data or the application installation script.

- *file_type*

  The file type of the user application data or the application installation script. This value can be one of the following:

  - `DATA`

    A `DATA` file type is a user application data file that is not executed but contains nonASCII data, such as static parameters.

  - `PRE_SCRIPT`

    A `PRE_SCRIPT` file type is an application installation script that is executed as part of the final stages of the Solaris JumpStart installation.

  - `POST_SCRIPT`

    A `POST_SCRIPT` file type is an application installation script that is executed after Solaris JumpStart installation at run level 2 or run level 3, that is, the `POST_SCRIPT` can be run before or after the startup of the Foundation Services. This is determined by the *run_level_value*.

- *file_location*

  The location of the user application data or the application installation script, expressed as a URI.

- *run_level_value*

The run-level for the execution of the application installation script, either `2` or `3`. The default value is `3`. The run level is only used for `POST_SCRIPT` application installation scripts.

- `next`

  This is an optional keyword that is used to determine the execution order of the `config` elements. For example:

  ```
  ELEMENT config softSwitch_3 next softSwitch_4
  ```

  In this case, the application will execute the `softSwitch_4` `config` element after the `softSwitch_3` `config` element.

EXAMPLE The following is an example of a user application configuration file `config.soft_switch`, for a node group `soft_switch`.

```
# Configuration of node group softSwitch - first stage -
#
# Script softSwitch_1.sh will be executed during Jumpstart finish stage
#ELEMENT config softSwitch_1
        INVOLVE file data_1_1
                file data_1_2
                file script_1_3

ELEMENT file data_1_1
        type DATA
        location 'file://config/softSwitch_1_1.conf'

ELEMENT file data_1_2
        type DATA
        location 'file://config/softSwitch_1_2.conf'

ELEMENT file script_1_3
        type PRE_SCRIPT
        location 'file://config/softSwitch_1.sh'

#
# Configuration of node group softSwitch - second stage -
#
# Script softSwitch_2.sh will be executed at init run-level 2
# The node will automatically reboot
#
ELEMENT config softSwitch_2
        INVOLVE file data_2_1
                file script_2_2

ELEMENT file data_2_1
        type DATA
        location 'file://config/softSwitch_2_1.conf'

ELEMENT file script_2_2
        type POST_SCRIPT
        runLevel 2
        location 'file://config/softSwitch_2.sh'

#
```

```
# Configuration of node group softSwitch - third stage -
#
# Scripts will be executed at init run-level 3 in the following order:
#    1. softSwitch_3.sh
#    2. softSwitch_4.sh
#
ELEMENT config softSwitch_3 next  softSwitch_4
        INVOLVE file script_3_1

ELEMENT file script_3_1
        type POST_SCRIPT
        location 'file://config/softSwitch_3.sh'

ELEMENT config softSwitch_4
        INVOLVE file script_4_1

ELEMENT file script_4_1
        type POST_SCRIPT
        location 'file://config/softSwitch_4.sh'
```

ATTRIBUTES See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWnhsmc
Interface Stability	Evolving

SEE ALSO `slconfig`(1M)

Devices

NAME

cgtp – CGTP virtual device driver

SYNOPSIS

/dev/cgtp

DESCRIPTION

The Carrier Grade Transport Protocol (CGTP) driver is a multi-threaded, loadable, clonable, STREAMS virtual device driver, compatible with the connectionless Data Link Provider Interface, `dlpi`(7P). CGTP is not related to any particular hardware and is purely virtual. Delivered with the Foundation Services, CGTP provides a single point of convergence for redundant incoming data flows. CGTP IP addresses are defined on each cluster node.

The CGTP does not send and receive data, but is instead normally used to indirectly accumulate incoming CGTP traffic from underlying redundant interfaces through internal IP routing. It supports most of the normal functions of a `DLPI` Ethernet device, the major exception being data transfer. All data packets (`DL_UNITDATA_REQ`) are rejected with an error of `network down`.

INTERFACE LEVEL

The cloning character-special device `/dev/cgtp` is used to register the Foundation Services cluster node's CGTP IP addresses.

The `cgtp` driver is a "style 2" Data Link Service provider. All messages of the `M_PROTO` and `M_PCPROTO` types are interpreted as `DLPI` primitives. Valid `DLPI` primitives are defined in the `sys/dlpi.h` file. For more information, see the `dlpi`(7P) man page. Send an explicit `DL_ATTACH_REQ` message to associate the opened stream with a particular device (`ppa`). As CGTP is purely a virtual driver, parameter configuration using `DL_INFO_REQ` is unnecessary.

The `ppa` ID is interpreted as an unsigned `long` data type and indicates the corresponding device instance (unit) number. An error (`DL_ERROR_ACK`) is returned by the driver if the `ppa` field value does not correspond to a valid device instance number for this system. The device is initialized on first attach and deinitialized (stopped) at last detach.

The values returned by the driver in the `DL_INFO_ACK` primitive in response to the `DL_INFO_REQ` from the user are as follows:

- The maximum service data unit (SDU) is 1500 (`ETHERMTU` - defined in `sys/ethernet.h`).
- The minimum SDU is 0.
- The `dlsap` address length is 8.
- The MAC type is `DL_ETHER`. MAC address is fixed to `0:0:0:0:0:0`.
- The `sap` length value is -2, meaning that the physical address component is followed immediately by a 2-byte `sap` component within the `DLSAP` address.
- No optional quality of service (`QOS`) support is included at present, so the `QOS` fields are 0.
- The provider style is `DL_STYLE`.
- The version is `DL_VERSION_2`.

- The broadcast address value is the Ethernet or IEEE broadcast address (0xFFFFFF).

Once in the DL_ATTACHED state, the user must send a DL_BIND_REQ to associate a particular SAP (Service Access Pointer) with the stream. The cgtp driver interprets the sap field within the DL_BIND_REQ as an Ethernet "type". Therefore, valid values for the sap field are in the [0-0xFFFF] range. Only one Ethernet type can be bound to the stream at any time.

The cgtp driver DLSAP address format consists of the 6-byte physical (Ethernet) address component followed immediately by the 2-byte sap (type) component producing an 8-byte DLSAP address. Applications must not hardcode to this particular implementation-specific DLSAP address format, but instead use information returned in the DL_INFO_ACK primitive to compose and decompose DLSAP addresses. The sap length, full DLSAP length, and sap or physical ordering are included within DL_INFO_ACK. The physical address length can be computed by subtracting the sap length from the full DLSAP address length or by issuing the DL_PHYS_ADDR_REQ to obtain the current physical address associated with the stream.

The DL_ENABMULTI_REQ and DL_DISABMULTI_REQ primitives enable or disable reception of individual multicast group addresses. A set of multicast addresses can be iteratively created and modified on a per-stream basis using these primitives. These primitives are accepted by the driver in any state following DL_ATTACHED.

The DL_PHYS_ADDR_REQ primitive returns the 6-octet Ethernet address currently associated (attached) to the stream in the DL_PHYS_ADDR_ACK primitive. This primitive is valid only in states following a successful DL_ATTACH_REQ.

The DL_SET_PHYS_ADDR_REQ primitive does not change the 6-octet Ethernet address currently associated (attached) to this stream.

After the cgtp virtual driver is installed, the administrator must configure the IP address with the address mask.

For example:

```
example% ifconfig cgtp0 plumb
example% ifconfig cgtp0 10.128.3.4/24 broadcast 10.128.3.255 up
example% ifconfig cgtp0
cgtp0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu \
1500 index 2
inet 10.128.3.4 netmask ffffff00 broadcast 10.128.3.255
ether 0:0:0:0:0:0
```

Aliasing is permitted on the cgtp driver. Configuring the cgtp virtual driver is not sufficient to access full CGTP functionalities. CGTP configuration must be done at the same time as CGTP routes configuration using the route command. In particular, there is a need to configure specific routes for broadcasts and multicasts by using the route command.

For instructions on how to manually install and configure CGTP on cluster nodes, see the *Netra High Availability Suite Foundation Services 2.1 6/03 Custom Installation Guide*. For instructions on how to manually install and configure CGTP on a standalone nodes, see the *Netra High Availability Suite Foundation Services 2.1 6/03 Standalone CGTP Guide*.

CGTP never actually sends or receives data. Instead, the CGTP configuration must be done at the same time as configuration of no more than two physical interfaces of any type that are responsible for redundancy.

When the `cgtp0` virtual physical interface and the two physical interfaces are properly configured, and all the necessary routes are correctly configured on the cluster, data will be sent and received through the redundant interfaces. At the emitter, `cgtp` is used (if specified by `route` command) to specify the source address of the CGTP IP packets. Packets are then transmitted on the physical interfaces. At the receiver, incoming CGTP packets are received by redundant interfaces, the duplicates are filtered out at the IP level, and the remaining packets converge in the `cgtp` stream to be presented to upper applications (after an optional reassembly).

Note – The use of IPv6 is not supported for use with CGTP.

Manually Configuring CGTP using Logical Interfaces

Logical interfaces can be configured only after the creation and configuration of physical interfaces. Two logical interfaces are configured on each node.

Additional logical interfaces can be created on a specified CGTP interface, provided the CGTP interface and the logical interfaces share the same subnet address. This means the primary CGTP address and the logical (or aliased) addresses must only differ by their host ID. Routes to the aliased interfaces can accordingly be set up on the other cluster nodes.

FILES

`/dev/cgtp` cgtp special character device.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	`SUNWnhtp8`, `SUNWnhtu8` for Solaris 8 2/02 `SUNWnhtp9`, `SUNWnhtu9` for Solaris 9
Interface Stability	Evolving

SEE ALSO

`dlpi`(7P)

Index

D

E

F

H

I

K

M

N

P

S

T

U