Sun Cluster Reference Manual for Solaris OS

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

Part No: 820–2756–10
December 2007, Revision A

Copyright 2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun^{TM} Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux Etats-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certaines composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, Java et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la legislation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la legislation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement designés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

070926@18741

Contents

Preface 9

Introduction 15
Intro(1CL) 16

SC32 1 23
libschost.so.1(1) 24

SC32 1cl 29
claccess(1CL) 30
cldevice(1CL) 35
cldevicegroup(1CL) 50
clinterconnect(1CL) 71
clnasdevice(1CL) 81
clnode(1CL) 93
clquorum(1CL) 110
clquorumserver(1CL) 123
clreslogicalhostname(1CL) 129
clresource(1CL) 148
clresourcegroup(1CL) 176
clresourcetype(1CL) 198
clressharedaddress(1CL) 210
clsetup(1CL) 229
clsnmphost(1CL) 231
clsnmpmib(1CL) 238
clsnmpuser(1CL) 246
cltelemetryattribute(1CL) 254

cluster(1CL) 269
clvxvm(1CL) 291

SC32 1ha 295
rt_callbacks(1HA) 296
scdsbuilder(1HA) 302
scdsconfig(1HA) 303
scdscreate(1HA) 306
scha_cluster_get(1HA) 309
scha_cmds(1HA) 312
scha_control(1HA) 320
scha_resource_get(1HA) 326
scha_resourcegroup_get(1HA) 332
scha_resource_setstatus(1HA) 335
scha_resourcetype_get(1HA) 338

SC32 1m 341
cconsole(1M) 342
ccp(1M) 344
chosts(1M) 346
cl_eventd(1M) 347
cports(1M) 348
halockrun(1M) 349
hatimerun(1M) 351
pmfadm(1M) 353
pnmd(1M) 358
rdt_setmtu(1M) 359
rpc.pmfd(1M) 360
sccheck(1M) 361
sccheckd(1M) 365
scconf(1M) 366
scconf_dg_rawdisk(1M) 386
scconf_dg_svm(1M) 389
scconf_dg_vxvm(1M) 391
scconf_quorum_dev_netapp_nas(1M) 394

scconf_quorum_dev_quorum_server(1M) 396
scconf_quorum_dev_scsi(1M) 398
scconf_transp_adap_bge(1M) 401
scconf_transp_adap_ce(1M) 402
scconf_transp_adap_e1000g(1M) 403
scconf_transp_adap_eri(1M) 404
scconf_transp_adap_ge(1M) 405
scconf_transp_adap_hme(1M) 406
scconf_transp_adap_ibd(1M) 407
scconf_transp_adap_qfe(1M) 408
scconf_transp_adap_sci(1M) 409
scconf_transp_jct_dolphinswitch(1M) 410
scconf_transp_jct_etherswitch(1M) 411
scconf_transp_jct_ibswitch(1M) 412
scdidadm(1M) 413
scdpm(1M) 422
sceventmib(1M) 426
scgdevs(1M) 432
scinstall(1M) 434
scnas(1M) 458
scnasdir(1M) 462
scprivipadm(1M) 465
scprivipd(1M) 469
scqsd(1M) 470
scrgadm(1M) 471
scsetup(1M) 481
scshutdown(1M) 483
scsnapshot(1M) 485
scstat(1M) 488
scswitch(1M) 493
sctelemetry(1M) 508
scversions(1M) 512
scvxinstall(1M) 514
sc_zonesd(1M) 519

SC32 3ha 521
scds_calls(3HA) 522
scds_close(3HA) 527
scds_error_string(3HA) 528
scds_failover_rg(3HA) 529
scds_fm_action(3HA) 530
scds_fm_net_connect(3HA) 533
scds_fm_net_disconnect(3HA) 536
scds_fm_print_probes(3HA) 538
scds_fm_sleep(3HA) 539
scds_fm_tcp_connect(3HA) 541
scds_fm_tcp_disconnect(3HA) 543
scds_fm_tcp_read(3HA) 544
scds_fm_tcp_write(3HA) 546
scds_free_ext_property(3HA) 548
scds_free_netaddr_list(3HA) 549
scds_free_net_list(3HA) 550
scds_free_port_list(3HA) 551
scds_get_ext_property(3HA) 552
scds_get_netaddr_list(3HA) 554
scds_get_port_list(3HA) 556
scds_get_resource_group_name(3HA) 557
scds_get_resource_name(3HA) 558
scds_get_resource_type_name(3HA) 559
scds_get_rg_hostnames(3HA) 560
scds_get_rs_hostnames(3HA) 561
scds_get_zone_name(3HA) 562
scds_hasp_check(3HA) 564
scds_initialize(3HA) 566
scds_pmf_get_status(3HA) 568
scds_pmf_restart_fm(3HA) 570
scds_pmf_signal(3HA) 571
scds_pmf_start(3HA) 573
scds_pmf_stop(3HA) 575
scds_pmf_stop_monitoring(3HA) 577
scds_print_netaddr_list(3HA) 579

scds_print_net_list(3HA) 580
scds_print_port_list(3HA) 581
scds_property_functions(3HA) 582
scds_restart_resource(3HA) 588
scds_restart_rg(3HA) 589
scds_simple_net_probe(3HA) 590
scds_simple_probe(3HA) 592
scds_svc_wait(3HA) 594
scds_syslog(3HA) 597
scds_syslog_debug(3HA) 598
scds_timerun(3HA) 600
scha_calls(3HA) 602
scha_cluster_getlogfacility(3HA) 608
scha_cluster_getnodename(3HA) 610
scha_cluster_getzone(3HA) 611
scha_cluster_open(3HA) 613
scha_control(3HA) 618
scha_resourcegroup_open(3HA) 624
scha_resource_open(3HA) 628
scha_resource_setstatus(3HA) 636
scha_resourcetype_open(3HA) 638
scha_strerror(3HA) 642

SC32 4 645
clusters(4) 646
commandlog(4) 648
rt_reg(4) 650
scqsd.conf(4) 658
serialports(4) 660

SC32 5 663
property_attributes(5) 664
rg_properties(5) 666
r_properties(5) 679
rt_properties(5) 699

scalable_service(5) 709
SUNW.crs_framework(5) 711
SUNW.derby(5) 714
SUNW.Event(5) 716
SUNW.gds(5) 721
SUNW.HADerby(5) 728
SUNW.HAStoragePlus(5) 730
SUNW.Proxy_SMF_failover(5) 734
SUNW.Proxy_SMF_loadbalanced(5) 737
SUNW.Proxy_SMF_multimaster(5) 740
SUNW.rac_cvm(5) 743
SUNW.rac_framework(5) 747
SUNW.rac_svm(5) 749
SUNW.rac_udlm(5) 752
SUNW.ScalDeviceGroup(5) 756
SUNW.ScalMountPoint(5) 761
SUNW.SCTelemetry(5) 767

SC32 5cl 769
clconfiguration(5CL) 770

SC32 7 795
clprivnet(7) 796
did(7) 797

SC32 7p 799
sctransp_dlpi(7p) 800

Index 801

Preface

The *Sun Cluster Reference Manual* provides reference information for commands, functions, and other public interfaces in Sun™ Cluster software. This book is intended for experienced system administrators with extensive knowledge of Sun software and hardware. This book is not to be used as a planning or presales guide. The information in this book assumes knowledge of the Solaris™ Operating System and expertise with the volume manager software that is used with Sun Cluster software.

Both novice users and those familiar with the Solaris Operating System can use online man pages to obtain information about their SPARC™ based system or x86 based system and its features.

A man page is intended to answer concisely the question "What does this command do?" The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Note – Sun Cluster software runs on two platforms, SPARC and x86. The information in this book pertains to both platforms unless otherwise specified in a special chapter, section, note, bulleted item, figure, table, or example.

Overview

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

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1CL describes, in alphabetical order, commands that are used for the maintenance and administration of Sun Cluster.
- Section 1HA describes describes, in alphabetical order, Sun Cluster high availability (HA) commands.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 3HA describes, in alphabetical order, Sun Cluster HA and data services functions.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.

- Section 5 contains miscellaneous Sun Cluster documentation such as descriptions of resource types.
- Section 5CL describes Sun Cluster standards, environments, and macros.
- Section 7 describes Sun Cluster device and network interfaces.
- Section 7P describes Sun Cluster protocols.

The following is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if no bugs can be reported, no BUGS section is included. See the `intro` pages for more information and detail about each section, and `man`(1) for general information about man pages.

NAME — This section gives the names of the commands or functions that are documented, followed by a brief description of what they do.

SYNOPSIS — This section shows the syntax of commands or functions. If 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 that is 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. All characters within braces must be treated as a unit.

PROTOCOL — This section occurs only in subsection 3R and indicates the protocol description file.

DESCRIPTION	This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. DESCRIPTION does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, and functions are described under USAGE.
IOCTL	This section appears on pages in Section 7 only. Only the device class that supplies appropriate parameters to the `ioctl`(2) system call is called `ioctl` and generates its own heading. `ioctl` calls for a specific device are listed alphabetically (on the man page for that specific device). `ioctl` calls are used for a particular class of devices. All these calls have an `io` ending, such as `mtio`(7I).
OPTIONS	This section 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 that are 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` that indicates 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 that are 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, which includes 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 are returned for various error conditions.
FILES	This section lists all file names that are referred to by the man page, files of interest, and files created or required by commands. Each file name 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 that caused the error.

WARNINGS	This section lists warnings about special conditions that could seriously affect your working conditions. WARNINGS is not a list of diagnostics.
NOTES	This section lists additional information that does not belong anywhere else on the page. NOTES covers points of special interest to the user. Critical information is never covered here.
BUGS	This section describes known bugs and, wherever possible, suggests workarounds.

REFERENCE

Introduction

Name Intro – introduction to Sun Cluster maintenance commands

Description This section describes the object-oriented command set for Sun Cluster. Although the original Sun Cluster command set is still available, use the object-oriented commands for more intuitive configuration of your cluster. In addition, future new features might not be available in the original command set.

The object-oriented command set uses a common prefix `cl`. The original command set used the prefix `sc`. Both the `sc` and `cl` commands are located in `/usr/cluster/bin`.

Many commands in this command set have both a long form and a short form. For example, clresource(1CL) and clrs(1CL) are identical.

Each object-oriented command is designed to manage a single type of cluster object. The command name indicates the type of object that it manages. For example, the `clresource` command manages Sun Cluster data service resources. Within a command, subcommands define operations that are allowed on the specific cluster object.

The general form of commands in the object-oriented command set is as follows:

cmdname [*subcommand*] [*option*…] [*operand* …]

Options that you use with the object-oriented commands also have a long form and a short form. You specify the short form of an option with a single dash (`-`) followed by a single character. You specify the long form of an option with two dashes (`--`) followed by an option word. For example, `-p` is the short form of the property option. `--property` is the long form.

Some options accept an option argument while others do not. If an option accepts an option argument, the option argument is required. The `-?` option requires no arguments. However, the `--property` option requires an option argument that identifies the property being operated on.

You can group the short form of options without arguments behind a single dash (`-`). For example, `-emM`. You must separate groups of option-arguments following an option either by commas, or by a tab or a space character. When using a tab or space, surround the option-arguments with quotation marks (-o xxx,z,yy or -o “xxx z yy”).

To specify option arguments with long option names, use either the `--input=`*configurationfile* format or the `--input` *configurationfile* format.

All commands in this command set accept the `-?` or `--help` option. If you provide these options without a subcommand, summary help for the command is displayed. If you provide a subcommand, help for that subcommand only is displayed.

Certain commands work in conjunction with a configuration file. For information on the required format of this file, see the clconfiguration(5CL) man page.

Many subcommands in this command set accept + as an operand to indicate all applicable objects.

List Of Commands This section describes, in alphabetical order, the object-oriented commands that are available with the Sun Cluster product.

claccess(1CL)
: Manage Sun Cluster access policies for adding nodes

cldevice(1CL), cldev(1CL)
: Manage Sun Cluster devices

cldevicegroup(1CL), cldg(1CL)
: Manage Sun Cluster device groups

clinterconnect(1CL), clintr(1CL)
: Manage the Sun Cluster interconnect

clnasdevice(1CL), clnas(1CL)
: Manage access to NAS devices for Sun Cluster

clnode(1CL)
: Manage Sun Cluster nodes

clquorum(1CL), clq(1CL)
: Manage Sun Cluster quorum

clreslogicalhostname(1CL), clrslh(1CL)
: Manage Sun Cluster resources for logical host names

clresource(1CL), clrs(1CL)
: Manage resources for Sun Cluster data services

clresourcegroup(1CL), clrg(1CL)
: Manage resource groups for Sun Cluster data services

clresourcetype(1CL), clrt(1CL)
: Manage resource types for Sun Cluster data services

clressharedaddress(1CL), clrssa(1CL)
: Manage Sun Cluster resources for shared addresses

clsetup(1CL)
: Configure Sun Cluster interactively

clsnmphost(1CL)
: Administer Sun Cluster SNMP hosts

clsnmpmib(1CL), clmib(1CL)
: Administer Sun Cluster SNMP MIB

clsnmpuser(1CL)
: Administer Sun Cluster SNMP users

cltelemetryattribute(1CL)
: Configure system resource monitoring.

cluster(1CL)
: Manage the global configuration and the global status of Sun Cluster

clvxvm(1CL)
: Configure Veritas Volume Manager for Sun Cluster

Mapping Original Sun Cluster Commands to Object-Oriented Commands

Because the newer command set is object oriented, a clear one-to-one mapping from the original command set does not exist. The following list provides some common Sun Cluster commands from the original set and their object-oriented set equivalents.

`scstat`
: `cluster status`

  You can also use the `status` subcommands that are available with many of the object-oriented commands.

`scinstall`
: Use `cluster create` to create a cluster from an XML configuration file.

  To create a cluster interactively, use `scinstall`.

`scrgadm`
: - `clresource`
  - `clresourcetype`
  - `clresourcegroup`

  `clressharedaddress` and `clreslogicalhostname` provide additional conveniences when you work with those particular resource types.

`scswitch`
: - `clresource`
  - `clresourcetype`
  - `clresourcegroup`
  - `clreslogicalhostname`
  - `clressharedaddress`
  - `clnode evacuate` (to move off a node all resource groups and device groups)

`scconf`
: - `cldevicegroup`
  - `clinterconnect`
  - `clquorum`
  - `clnode`
  - `claccess`

  Use `cluster show` instead of `scconf -p`.

`scdidadm`	`cldevice`
`scgdevs`	`cldevice populate`
`scdpm`	`cldevice`
`scvxinstall`	`clvxvm`
`scnas`, `scnasdir`	`clnasdevice`
`scsetup`	`clsetup`

Exit Status If an object-oriented Sun Cluster command is successful for all specified operands, the command returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

These exit codes are shared across this set of commands.

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`5` `CL_ERECONF`
Cluster is reconfiguring

The cluster is reconfiguring.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`9` `CL_ESTATE`
Object is in wrong state

You tried to modify a property, a resource group, or other object that you cannot modify at that particular time or at any time.

10 CL_EMETHOD
Resource method failed

A method of a resource failed. The method failed for one of the following reasons:

- The `validate` method failed when you tried to create a resource or modify the properties of a resource.
- A method other than `validate` failed when you tried to enable, disable, or delete a resource.

15 CL_EPROP
Invalid property

The property or value that you specified with the `-p`, `-y`, or `-x` option does not exist or is not allowed.

18 CL_EINTERNAL
Internal error was encountered

An internal error indicates a software defect or other defect.

35 CL_EIO
I/O error

A physical input/output error has occurred.

36 CL_ENOENT
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

37 CL_EOP
Operation not allowed

You tried to perform an operation on an unsupported configuration, or you performed an unsupported operation.

38 CL_EBUSY
Object busy

You attempted to remove a cable from the last cluster interconnect path to an active cluster node. Or, you attempted to remove a node from a cluster configuration from which you have not removed references.

39 CL_EEXIST
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

41 CL_ETYPE
Invalid type

The type that you specified with the -t or -p option does not exist.

50 CL_ECLMODE
Node is in cluster mode

You attempted to perform an operation on a node that is booted in cluster mode. However, you can perform this operation only on a node that is booted in noncluster mode.

51 CL_ENOTCLMODE
Node is not in cluster mode

You attempted to perform an operation on a node that is booted in noncluster mode. However, you can perform this operation only on a node that is booted in cluster mode.

See Also getopt(1)

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

REFERENCE

SC32 1

Name libschost.so.1 – shared object to provide logical host name instead of a physical host name

Synopsis

```
libschost.so.1
```

Description The `libschost.so.1` shared object provides a mechanism by which the physical host name can be selectively configured for launched processes and their descendants.

In the Sun Cluster environment, an application might attempt to access the same host name after a failover or switchover. As a result, the failover or switchover fails because the name of the physical host changes after a failover or switchover. In such a scenario, the application data service can use the `libschost.so.1` shared object to provide a logical host name to the application rather than a physical host name.

To enable `libschost.so.1`, you need to set the `SC_LHOSTNAME` environment variable as well as the following two environment variables:

```
LD_PRELOAD_32=$LD_PRELOAD_32:/usr/cluster/lib/libschost.so.1
LD_PRELOAD_64=$LD_PRELOAD_64:/usr/cluster/lib/64/libschost.so.1
```

By setting both the `LD_PRELOAD_32` and `LD_PRELOAD_64` environment variables, you ensure that the `libschost.so.1` shared object works with both 32-bit and 64-bit applications.

Security The runtime linker accesses the default trusted directory `/usr/lib/secure` for 32-bit objects and `/usr/lib/secure/64` for 64-bit objects. If your secure applications use the `libschost.so.1` shared object, you need to ensure that the `libschost.so.1` shared object is accessed from a trusted directory.

To do so, create a symbolic link from `/usr/cluster/lib/libschost.so.1` to `/usr/lib/secure/libschost.so.1` for 32-bit applications or from `/usr/cluster/lib/64/libschost.so.1` to `/usr/lib/secure/64/libschost.so.1` for 64-bit applications.

After you create these symbolic links, the `LD_PRELOAD_32` and `LD_PRELOAD_64` environment variables use the `libschost.so.1` shared object from a trusted directory.

You can also use the `crle` command to specify additional trusted directories or to change the default trusted directory for secure applications. See the `crle`(1) man page.

Environment Variables Once preloaded, the `libschost.so.1` shared object reads the following environment variable and returns it as the host name.

`SC_LHOSTNAME=`*hostname*

`SC_LHOSTNAME` specifies the logical host name. The specified host name is available to all launched and descendant processes.

The *hostname* value can be a maximum of `MAXHOSTNAMELEN` characters long. The `MAXHOSTNAMELEN` constant is defined as 256 characters in the `netdb.h` header file.

Examples

EXAMPLE 1 Configuring a Logical Host Name With a Logical Host Name at Runtime in C

The C code in the following example configures a host name with a logical host name. This example includes a call to the `scds_get_rs_hostnames()` Sun Cluster function and includes references to the `scds_handle_t` and `scds_net_resource_list_t` Sun Cluster data structures.

The `scds_get_rs_hostnames()` function provides a list of host names that are used by a resource. The code assigns the first host name value in this list to the `SC_LHOSTNAME` environment variable.

Any application that starts after you execute the following code gets a logical host name rather than a physical host name.

```
/* 13 bytes to hold "SC_LHOSTNAME=" string */
#define HOSTLENGTH (MAXHOSTNAMELEN + 13)

/* 14 bytes to hold "LD_PRELOAD_XX=" string */
#define PATHLENGTH (MAXPATHLEN + 14)

char lhostname[HOSTLENGTH], ld_32[PATHLENGTH], \
     ld_64[PATHLENGTH];

scds_get_rs_hostnames(scds_handle, &snrlp);
if (snrlp != NULL && snrlp->num_netresources != 0) {
    snprintf(lhostname, HOSTLENGTH, "SC_LHOSTNAME=%s", \
        snrlp->netresources[0].hostnames[0]);
    putenv(lhostname);
}

/* Setting LD_PRELOAD_32 environment variable */
if (getenv("LD_PRELOAD_32") == NULL)
    snprintf(ld_32, PATHLENGTH, "LD_PRELOAD_32="
        "/usr/cluster/lib/libschost.so.1");
else
    snprintf(ld_32, PATHLENGTH, "LD_PRELOAD_32=%s:"
        "/usr/cluster/lib/libschost.so.1", \
         getenv("LD_PRELOAD_32"));

putenv(ld_32);

/* Setting LD_PRELOAD_64 environment variable */
if (getenv("LD_PRELOAD_64") == NULL)
    snprintf(ld_64, PATHLENGTH, "LD_PRELOAD_64="
        "/usr/cluster/lib/64/libschost.so.1");
else
    snprintf(ld_64, PATHLENGTH,
        "LD_PRELOAD_64=%s:/usr/cluster/lib/"
        "64/libschost.so.1", getenv("LD_PRELOAD_64"));
```

EXAMPLE 1 Configuring a Logical Host Name With a Logical Host Name at Runtime in C *(Continued)*

```
    putenv(ld_64);
```

EXAMPLE 2 Configuring a Logical Host Name With a Logical Host Name at Runtime With Shell Commands

The shell commands in the following example show how an application data service configures a host name with a logical host name by using the `gethostnames` command. The `gethostnames` command takes the following arguments:

- `-R` *resource-name*
- `-G` *resourcegroup-name*
- `-T` *resourcetype-name*

The `gethostnames` command returns all the logical host names that are associated with that resource, separated by a semicolon (;). The commands assign the first host name value in this list to the `SC_LHOSTNAME` environment variable.

```
phys-schost-1$ LD_PRELOAD_32=$LD_PRELOAD_32:/usr/cluster/lib/libschost.so.1
phys-schost-1$ LD_PRELOAD_64=$LD_PRELOAD_64:/usr/cluster/lib/64/libschost.so.1
phys-schost-1$ SC_LHOSTNAME=‘/usr/cluster/lib/scdsbuilder/src/scripts/gethostnames \
               -R nfs-r -G nfs-rg -T SUNW.nfs:3.1 |cut -f1 -d","‘
phys-schost-1$ export LD_PRELOAD_32 LD_PRELOAD_64 SC_LHOSTNAME
```

EXAMPLE 3 Configuring a Logical Host Name for Secure Applications With Shell Commands

The shell commands in the following example configure the logical host name. Any secure application that starts after you execute the following shell commands gets the value of the `SC_LHOSTNAME` environment variable (that is, a logical host name) rather than a physical host name.

```
phys-schost-1$ cd /usr/lib/secure
phys-schost-1$ ln -s /usr/cluster/lib/libschost.so.1 .
phys-schost-1$ cd /usr/lib/secure/64
phys-schost-1$ ln -s /usr/cluster/lib/64/libschost.so.1 .
phys-schost-1$ LD_PRELOAD_32=$LD_PRELOAD_32:/usr/lib/secure/libschost.so.1
phys-schost-1$ LD_PRELOAD_64=$LD_PRELOAD_64:/usr/lib/secure/64/libschost.so.1
phys-schost-1$ SC_LHOSTNAME=test
phys-schost-1$ export LD_PRELOAD_32 LD_PRELOAD_64 SC_LHOSTNAME
```

Files `/usr/cluster/lib/libschost.so.1`
Default location of the shared object for 32-bit applications

`/usr/cluster/lib/64/libschost.so.1`
Default location of the shared object for 64-bit applications

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also `crle`(1), `cut`(1), `hostname`(1), `ld`(1), `ld.so.1`(1), `proc`(1), `uname`(1), `exec`(2), `sysinfo`(2), `uname`(2), `gethostname`(3C), `putenv`(3C), `snprintf`(3C), `system`(3C), `proc`(4)

Notes The logical host name is inherited.

User programs that fetch a host name by calling the following commands or functions can obtain a logical host name rather than a physical host name:

- `hostname` command
- `uname` command
- `uname()` function
- `sysinfo()` function
- `gethostname()` function

User programs that fetch a host name by other commands or functions cannot obtain a logical host name.

REFERENCE

SC32 1cl

Name claccess – manage Sun Cluster access policies for nodes

Synopsis

```
/usr/cluster/bin/claccess -V

/usr/cluster/bin/claccess [subcommand]
 -?

/usr/cluster/bin/claccess subcommand
 [options] -v [hostname[,...]]

/usr/cluster/bin/claccess allow  -h hostname[,...]

/usr/cluster/bin/claccess allow-all

/usr/cluster/bin/claccess deny  -h hostname[,...]

/usr/cluster/bin/claccess deny-all

/usr/cluster/bin/claccess list

/usr/cluster/bin/claccess set  -p protocol=authprotocol

/usr/cluster/bin/claccess show
```

Description The `claccess` command controls the network access policies for machines that attempt to access the cluster configuration. The `claccess` command has no short form.

The cluster maintains a list of machines that can access the cluster configuration. The cluster also stores the name of the authentication protocol that is used for these nodes to access the cluster configuration.

When a machine attempts to access the cluster configuration, for example when it asks to be added to the cluster configuration (see clnode(1CL)), the cluster checks this list to determine whether the node has access permission. If the node has permission, the node is authenticated and allowed access to the cluster configuration.

You can use the `claccess` command for the following tasks:

- To allow any new machines to add themselves to the cluster configuration and remove themselves from the cluster configuration
- To prevent any nodes from adding themselves to the cluster configuration and removing themselves from the cluster configuration
- To control the authentication type to check

You can use this command only in the global zone.

The general form of the `claccess` command is as follows:

`claccess` [*subcommand*] [*options*]

You can omit *subcommand* only if *options* specifies the `-?` option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are provided with the description of the option in the “OPTIONS” section of this man page.

Subcommands The following subcommands are supported:

`allow`
Allows the specified machine or machines to access the cluster configuration.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand. See `rbac`(5).

See also the description of the `deny` and the `allow-all` subcommands.

`allow-all`
Allows all machines to add themselves to access the cluster configuration.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See `rbac`(5).

See also the description of the `deny-all` and the `allow` subcommands.

`deny`
Prevents the specified machine or machines from accessing the cluster configuration.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See `rbac`(5).

See also the description of the `allow` and the `deny-all` subcommands.

`deny-all`
Prevents all machines from accessing the cluster configuration.

No access for any node is the default setting after the cluster is configured the first time.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See `rbac`(5).

See also the description of the `allow-all` and the `deny` subcommands.

`list`
Displays the names of the machines that have authorization to access the cluster configuration. To see the authentication protocol as well, use the `show` subcommand.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See `rbac`(5).

`set`
Sets the authentication protocol to the value that you specify with the `-p` option. By default, the system uses `sys` as the authentication protocol. See the `-p` option in “OPTIONS”.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See `rbac`(5).

`show`
Displays the names of the machines that have permission to access the cluster configuration. Also displays the authentication protocol.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See `rbac`(5).

Options The following options are supported:

`-?`
`--help`
Displays help information. When you use this option, no other processing is performed.

You can specify this option without a subcommand or with a subcommand. If you specify this option without a subcommand, the list of subcommands of this command is displayed. If you specify this option with a subcommand, the usage options for the subcommand are displayed.

`-h` *hostname*
`--host=`*hostname*
`--host` *hostname*
Specifies the name of the node being granted or denied access.

`-p` protocol=*authprotocol*
`--authprotocol=`*authentication_protocol*
`--authprotocol` *authentication_protocol*
Specifies the authentication protocol that is used to check whether a machine has access to the cluster configuration.

Supported protocols are `des` and `sys` (or `unix`). The default authentication type is `sys`, which provides the least amount of secure authentication. For more information on adding and removing nodes, see "Adding a Cluster Node" in *Sun Cluster System Administration Guide for Solaris OS*. For more information on these authentication types, see Chapter 16, "Using Authentication Services (Tasks)," in *System Administration Guide: Security Services*.

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The `-V` option displays only the version of the command. No other processing is performed.

`-v`
`--verbose`
Displays verbose information to standard output (`stdout`).

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

The following exit codes can be returned:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`18` `CL_EINTERNAL`
Internal error was encountered

An internal error indicates a software defect or other defect.

`39` `CL_EEXIST`
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

Examples **EXAMPLE 1** Allow a New Host Access

The following `claccess` command allows a new host to access the cluster configuration.

```
# claccess allow -h phys-schost-1
```

EXAMPLE 2 Set the Authentication Type

The following `claccess` command sets the current authentication type to `des`.

```
# claccess set -p protocol=des
```

EXAMPLE 3 Deny Access to All Hosts

The following `claccess` command denies all hosts access to the cluster configuration.

```
# claccess deny-all
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), clnode(1CL), cluster(1CL)

Notes

The superuser user can run all forms of this command.

Any user can run this command with the following subcommands and options:

- `-?` option
- `-V` option

To run this command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`allow`	`solaris.cluster.modify`
`allow-all`	`solaris.cluster.modify`
`deny`	`solaris.cluster.modify`
`deny-all`	`solaris.cluster.modify`
`list`	`solaris.cluster.read`
`set`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`

Name cldevice, cldev – manage Sun Cluster devices

Synopsis

```
/usr/cluster/bin/cldevice  -V

/usr/cluster/bin/cldevice  [subcommand] -?

/usr/cluster/bin/cldevice  subcommand  [options]
 -v [+ |  device ...]

/usr/cluster/bin/cldevice check  [-n node[,...]]
 [+]

/usr/cluster/bin/cldevice clear  [-n node[,...]]
 [+]

/usr/cluster/bin/cldevice combine  -t replication-type
 -g replication-device-group -d destination-device
 device

/usr/cluster/bin/cldevice export  [-o {- |  configfile}]
 [-n node[,...]] [+ |  device ...]

/usr/cluster/bin/cldevice list  [-n node[,...]]
 [+ |  device ...]

/usr/cluster/bin/cldevice monitor  [-i {- |  clconfigfile} ]
 [-n node[,...] ] {+ |  disk-device ...}

/usr/cluster/bin/cldevice populate

/usr/cluster/bin/cldevice refresh  [-n node[,...]]
 [+]

/usr/cluster/bin/cldevice rename  -d destination-device
 device

/usr/cluster/bin/cldevice repair  [-n node[,...]]
 {+ |  device ...}

/usr/cluster/bin/cldevice replicate  -t replication-type [-S source-node]
 -D destination-node [+]

/usr/cluster/bin/cldevice set
 -p default_fencing={global | pathcount | scsi3}
 [-n node[,...]] device ...

/usr/cluster/bin/cldevice show  [-n node[,...]]
 [+ |  device ...]

/usr/cluster/bin/cldevice status  [-s state] [-n node[,...]]
 [+ |  [disk-device ]]

/usr/cluster/bin/cldevice unmonitor  [-i {- |  clconfigfile} ]
 [-n node[,...]] {+ |  disk-device ...}
```

Description The `cldevice` command manages devices in the Sun Cluster environment. Use this command to administer the Sun Cluster device identifier (DID) pseudo device driver and to monitor disk device paths.

- The DID driver provides a device with a unique device ID, even if multiple paths to the device are available. See the did(7) man page for more information.
- A disk path is the connection between a cluster node and a physical disk or LUN storage device. The disk path includes the Solaris kernel driver stack, Host Bus Adapter, and any intervening cables, switches, or network connectivity.

The `cldev` command is the short form of the `cldevice` command. You can use either form of the command.

With the exception of the `list` and `show` subcommands, you must run the `cldevice` command from a cluster node that is online and in cluster mode.

The general form of this command is as follows:

`cldevice` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the `-?` option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are given with the description of the option in the OPTIONS section of this man page.

See the Intro(1CL)) man page for more information.

You can use this command only in the global zone.

Subcommands The following subcommands are supported:

`check`
Performs a consistency check to compare the kernel representation of the devices against the physical devices. On failing a consistency check, an error message is displayed. The process continues until all devices are checked.

By default, this subcommand affects only the current node. Use the `-n` option to perform the check operation for devices that are attached to another node.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`clear`
Removes all DID references to underlying devices that are no longer attached to the current node.

By default, this subcommand affects only the current node. Use the `-n` option to specify another cluster node on which to perform the clear operation.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`combine`
Combines the specified device with the specified destination device.

The `combine` subcommand combines the path for the source device with the path for the destination device. This combined path results in a single DID instance number, which is the same as the DID instance number of the destination device.

You can use the `combine` subcommand to manually configure DID devices for storage-based replication. However, for TrueCopy replicated devices, automatic configuration by using the `replicate` subcommand is the best practice to configure replicated devices.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`export`
Exports configuration information for a cluster device.

If you specify a file name with the `-o` option, the configuration information is written to that new file. If you do not supply the `-o` option, the configuration information is written to standard output.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list`
Displays all device paths.

If you supply no operand, or if you supply the plus sign (+) operand, the report includes all devices.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`monitor`
Turns on monitoring for the specified disk paths.

The `monitor` subcommand works only on disk devices. Tapes or other devices are not affected by this subcommand.

By default, this subcommand turns on monitoring for paths from all nodes.

Use the `-i` option to specify a cluster configuration file from which to set the monitor property of disk paths. The `-i` option starts disk-path monitoring on those disk paths that are marked in the specified file as monitored. No change is made for other disk paths. See the clconfiguration(5CL) man page for more information about the cluster configuration file.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`populate`
Populates the global-devices namespace.

The global-devices namespace is mounted under the `/global` directory. The namespace consists of a set of logical links to physical devices. Because the `/dev/global` directory is visible to each node of the cluster, each physical device is visible across the cluster. This visibility means that any disk, tape, or CD-ROM that is added to the global-devices namespace can be accessed from any node in the cluster.

The `populate` subcommand enables the administrator to attach new global devices to the global-devices namespace without requiring a system reboot. These devices might be tape drives, CD-ROM drives, or disk drives.

You must execute the `devfsadm`(1M) command before you run the `populate` subcommand. Alternatively, you can perform a reconfiguration reboot to rebuild the global-devices namespace and to attach new global devices. See the `boot`(1M) man page for more information about reconfiguration reboots.

You must run the `populate` subcommand from a node that is a current cluster member.

The `populate` subcommand performs its work on remote nodes asynchronously. Therefore, command completion on the node from which you issue the command does not signify that the command has completed operation on all cluster nodes.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`refresh`
Updates the device configuration information that is based on the current device trees on a cluster node. The command conducts a thorough search of the `rdsk` and `rmt` device trees. For each device identifier that was not previously recognized, the command assigns a new DID instance number. Also, a new path is added for each newly recognized device.

By default, this subcommand affects only the current node. Use the `-n` option with the `refresh` subcommand to specify the cluster node on which to perform the refresh operation.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`rename`
Moves the specified device to a new DID instance number.

The command removes DID device paths that correspond to the DID instance number of the source device and recreates the device path with the specified destination DID instance number. You can use this subcommand to restore a DID instance number that has been accidentally changed.

After you run the `rename` subcommand on all cluster nodes that are connected to the shared storage, run the `devfsadm` and `cldevice populate` commands to update the global-devices namespace with the configuration change.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`repair`
Performs a repair procedure on the specified device.

By default, this subcommand affects only the current node. Use the `-n` option to specify the cluster node on which to perform the repair operation.

If you supply no operand, or if you supply the plus sign (+) operand, the command updates configuration information on all devices that are connected to the current node.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`replicate`
Configures DID devices for use with storage-based replication.

The `replicate` subcommand combines each DID instance number on the source node with its corresponding DID instance number on the destination node. Each pair of replicated devices is merged into a single logical DID device.

By default, the current node is the source node. Use the `-S` option to specify a different source node.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`set`
Modifies the properties of the specified device.

Use the `-p` option to specify the property to modify.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`show`
Displays a configuration report for all specified device paths.

The report shows the paths to devices and whether the paths are monitored or unmonitored.

By default, the subcommand displays configuration information for all devices.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`status`
Displays the status of all specified disk-device paths.

By default, the subcommand displays the status of all disk paths from all nodes.

The `status` subcommand works only on disk devices. The report does not include tapes or other devices.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`unmonitor`
Turns off monitoring for the disk paths that are specified as operands to the command.

By default, the subcommand turns off monitoring for all paths from all nodes.

The `unmonitor` subcommand works only on disk devices. Tapes or other devices are not affected by this subcommand.

Use the `-i` option to specify a cluster configuration file from which to turn off monitoring for disk paths. Disk-path monitoring is turned off for those disk paths that are marked in the specified file as unmonitored. No change is made for other disk paths. See the clconfiguration(5CL) man page for more information.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

Options The following options are supported:

`-?`
`--help`
Displays help information.

This option can be used alone or with a subcommand.

- If you use this option alone, the list of available subcommands is printed.
- If you use this option with a subcommand, the usage options for that subcommand are printed.

When this option is used, no other processing is performed.

`-D` *destination-node*
`--destinationnode=`*destination-node*
`--destinationnode` *destination-node*
Specifies a destination node on which to replicate devices. You can specify a node either by its node name or by its node ID.

The `-D` option is only valid with the `replicate` subcommand.

`-d` *destination-device*
`--device=`*destination-device*
`--device` *destination-device*
Specifies the DID instance number of the destination device for storage-based replication.

Only use a DID instance number with the `-d` option. Do not use other forms of the DID name or the full UNIX path name to specify the destination device.

The `-d` option is only valid with the `rename` and `combine` subcommands.

`-g` *replication-device-group*
Specifies the replication device group. This option can be only be used with the `combine` subcommand.

`-i` {`-` | *clconfigfile*}
`--input={- |` *clconfigfile*`-}`
`--input {- |` *clconfigfile*`-}`
Specifies configuration information that is to be used for monitoring or unmonitoring disk paths. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through standard input. To specify standard input, specify the minus sign (`-`) instead of a file name.

The `-i` option is only valid with the `monitor` and `unmonitor` subcommands.

Options that you specify in the command override any options that are set in the configuration file. If configuration parameters are missing in the cluster configuration file, you must specify these parameters on the command line.

`-n` *node*[,…]
`--node=`*node*[,…]
`--node` *node*[,…]
Specifies that the subcommand includes only disk paths from nodes that are specified with the `-n` option. You can specify a node either by its node name or by its node ID.

`-o` {`-` | *configfile*}
`--output={- |` *configfile*`-}`
`--output {- |` *configfile*`-}`
Writes disk-path configuration information in the format that is defined by the clconfiguration(5CL) man page. This information can be written to a file or to standard output.

The `-o` option is only valid with the `export` subcommand.

If you supply a file name as the argument to this option, the command creates a new file and the configuration is printed to that file. If a file of the same name already exists, the command exits with an error. No change is made to the existing file.

If you supply the minus sign (-) as the argument to this option, the command displays the configuration information to standard output. All other standard output for the command is suppressed.

`-p default_fencing={global | pathcount | scsi3}`
`--property=default_fencing={global | pathcount | scsi3}`
`--property default_fencing={global | pathcount | scsi3}`
Specifies the property to modify.

Use this option with the `set` subcommand to modify the following property:

`default_fencing`
Overrides the global default fencing algorithm for the specified device.

Note – You cannot change the default fencing algorithm on a device that is configured as a quorum device.

The default fencing algorithm for a device can be set to one of the following values:

`global` Uses the global default fencing setting. See the cluster(1CL) man page for information about setting the global default for fencing.

`pathcount` Determines the fencing protocol by the number of DID paths that are attached to the shared device.

- For a device that uses fewer than three DID paths, the command sets the SCSI-2 protocol.
- For a device that uses three or more DID paths, the command sets the SCSI-3 protocol

`scsi3` Sets the SCSI-3 protocol. If the device does not support the SCSI-3 protocol, the fencing protocol setting remains unchanged.

`-S` *source-node*
`--sourcenode=`*source-node*
`--sourcenode` *source-node*
Specifies the source node from which devices are replicated to a destination node. You can specify a node either by its node name or by its node ID.

The `-S` option is only valid with the `replicate` subcommand.

`-s` *state*[,…]
`--state=`*state*[,…]
`--state` *state*[,…]
Displays status information for disk paths that are in the specified state.

The `-s` option is only valid with the `status` subcommand. When you supply the `-s` option, the status output is restricted to disk paths that are in the specified *state*. The following are the possible values of the *state*:

- `fail`
- `ok`
- `unknown`
- `unmonitored`

`-t`
Specifies the replication device type. This option can be used with the `replicate` and `combine` subcommands.

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommand, operands, or other options are ignored. The `-V` option only displays the version of the command. No other operations are performed.

`-v`
`--verbose`
Displays verbose information to standard output.

You can specify this option with any form of this command.

Operands The following operands are supported:

device
Specifies the name of a device. The device can be, but is not limited to, disks, tapes, and CD-ROMs.

If the subcommand accepts more than one device, you can use the plus sign (+) to specify all devices.

All subcommands of the `cldevice` command except the `repair` subcommand accept device paths as operands. The `repair` subcommand accepts only device names as operands. The *device* name can be either the full global path name, the device name, or the DID instance number. Examples of these forms of a device name are `/dev/did/dsk/d3`, `d3`, and `3`, respectively. See the did(7) man page for more information.

The device name can also be the full UNIX path name, such as`/dev/rdsk/c0t0d0s0`.

A specified device can have multiple paths that connect the device to nodes. If the `-n` option is not used, all paths from all nodes to the specified device are selected.

The `monitor`, `unmonitor`, and `status` subcommands only accept disk devices as operands.

Exit Status The complete set of exit status codes for all commands in this command set are listed on the Intro(1CL) man page.

If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

`1` `CL_ENOMEM`
Not enough swap space

`3` `CL_EINVAL`
Invalid argument

`6` `CL_EACCESS`
Permission denied

`9` `CL_ESTATE`
Object is in wrong state

`15` `CL_EPROP`
Invalid property

`35` `CL_EIO`
I/O error

`36` `CL_ENOENT`
No such object

`37` `CL_EOP`
Operation not allowed

Examples

EXAMPLE 1 Monitoring All Disk Paths in the Cluster

The following example shows how to enable the monitoring of all disk paths that are in the cluster infrastructure.

```
# cldevice monitor +
```

EXAMPLE 2 Monitoring a Single Disk Path

The following example shows how to enable the monitoring of the path to the disk `/dev/did/dsk/d3` on all nodes where this path is valid.

```
# cldevice monitor /dev/did/dsk/d3
```

EXAMPLE 3 Monitoring a Disk Path on a Single Node

The following examples show how to enable the monitoring of the path to the disks `/dev/did/dsk/d4` and `/dev/did/dsk/d5` on the node `phys-schost-2`.

EXAMPLE 3 Monitoring a Disk Path on a Single Node *(Continued)*

The first example uses the `-n` option to limit monitoring to disk paths that are connected to the node `phys-schost-2`, then further limits monitoring to the specified devices `d4` and `d5`.

```
# cldevice monitor -n phys-schost-2 d4 d5
```

The second example specifies the disk paths to monitor by their *node*:*device* names, `phys-schost-2:d4` and `phys-schost-2:d5`.

```
# cldevice monitor phys-schost-2:d4 phys-schost-2:d5
```

EXAMPLE 4 Printing All Disk Paths and Their Status

The following example shows how to print all disk paths in the cluster and their status.

```
# cldevice status
Device Instance             Node               Status
---------------             ----               ------
/dev/did/rdsk/d1            phys-schost-2      Unmonitored

/dev/did/rdsk/d2            phys-schost-2      Unmonitored

/dev/did/rdsk/d3            phys-schost-1      Ok
                            phys-schost-2      Ok

/dev/did/rdsk/d4            phys-schost-1      Ok
                            phys-schost-2      Ok

/dev/did/rdsk/d5            phys-schost-1      Unmonitored
```

EXAMPLE 5 Printing All Disk Paths That Have the Status `fail`

The following example shows how to print all disk paths that are monitored on the node `phys-schost-2` and that have the status `fail`.

```
# cldevice status -s fail -n phys-schost-1
Device Instance             Node               Status
---------------             ----               ------
/dev/did/rdsk/d3            phys-schost-1      Fail

/dev/did/rdsk/d4            phys-schost-1      Fail
```

EXAMPLE 6 Printing the Status of All Disk Paths From a Single Node

The following example shows how to print the path and the status for all disk paths that are online on the node `phys-schost-2`.

EXAMPLE 6 Printing the Status of All Disk Paths From a Single Node *(Continued)*

```
# cldevice status -n phys-schost-1
Device Instance             Node               Status
---------------             ----               ------
/dev/did/rdsk/d3            phys-schost-1      Ok

/dev/did/rdsk/d4            phys-schost-1      Ok

/dev/did/rdsk/d5            phys-schost-1      Unmonitored
```

EXAMPLE 7 Adding New Devices to the Device Configuration Database

The following example shows how to update the CCR database with the current device configurations for the node `phys-schost-2`, from which the command is issued. This command does not update the database for devices that are attached to any other node in the cluster.

```
phys-schost-2# cldevice refresh
```

EXAMPLE 8 Combining Devices Under a Single DID

The following example shows how to combine the path for one device with the path for another device. This combined path results in a single DID instance number, which is the same as the DID instance number of the destination device.

```
# cldevice combine -t srdf -g devgrp1 -d 20 30
```

EXAMPLE 9 Listing the Device Paths For a Device Instance

The following example shows how to list the paths for all devices that correspond to instance `3` of the DID driver.

```
# cldevice list 3
d3
```

EXAMPLE 10 Listing all Device Paths in the Cluster

The following example shows how to list all device paths for all devices that are connected to any cluster node.

```
# cldevice list -v
DID Device          Full Device Path
----------          ----------------
d1                  phys-schost-1:/dev/rdsk/c0t0d0
d2                  phys-schost-1:/dev/rdsk/c0t1d0
```

EXAMPLE 10 Listing all Device Paths in the Cluster *(Continued)*

```
d3                  phys-schost-1:/dev/rdsk/c1t8d0
d3                  phys-schost-2:/dev/rdsk/c1t8d0
d4                  phys-schost-1:/dev/rdsk/c1t9d0
d4                  phys-schost-2:/dev/rdsk/c1t9d0
d5                  phys-schost-1:/dev/rdsk/c1t10d0
d5                  phys-schost-2:/dev/rdsk/c1t10d0
d6                  phys-schost-1:/dev/rdsk/c1t11d0
d6                  phys-schost-2:/dev/rdsk/c1t11d0
d7                  phys-schost-2:/dev/rdsk/c0t0d0
d8                  phys-schost-2:/dev/rdsk/c0t1d0
```

EXAMPLE 11 Configuring Devices for Use With Storage-Based Replication

The following example configures DID devices for use with storage-based replication. The command is run from the source node, which is configured with replicated devices. Each DID instance number on the source node are combined with its corresponding DID instance number on the destination node, `phys-schost-1`.

```
# cldevice replicate -t truecopy -D phys-schost-1
```

EXAMPLE 12 Setting the SCSI Protocol of a Single Device

The following example sets the device `11`, specified by instance number, to the SCSI-3 protocol. This device is not a configured quorum device.

```
# cldevice set -p default_fencing=scsi3 11
```

EXAMPLE 13 Performing a Repair Procedure By Using the Device Name

The following example shows how to perform a repair procedure on the device identifier that was associated with the device `/dev/dsk/c1t4d0`. This device was replaced with a new device to which a new device identifier is now associated. In the database, the `repair` subcommand records that instance number now corresponds to the new device identifier.

```
# cldevice repair c1t4d0
```

EXAMPLE 14 Performing a Repair Procedure By Using the Instance Number

The following example shows how to provide an alternate method to perform a repair procedure on a device identifier. This example specifies the instance number that is associated with the device path to the replaced device. The instance number for the replaced device is `2`.

```
# cldevice repair 2
```

EXAMPLE 15 Populating the Global-Devices Namespace

The following example shows how to populate the global-devices namespace after adding new global devices or moving a DID device to a new instance number.

```
# devfsadm
# cldevice populate
```

EXAMPLE 16 Moving a DID Device

The following example moves the DID instance on the source instance, `15`, to a new DID instance, `10`, then updates the global-devices namespace with the configuration change.

```
# cldevice rename 15:10
# devfsadm
# cldevice populate
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cluster(1CL), `boot`(1M), `devfsadm`(1M), clconfiguration(5CL), `rbac`(5), did(7)

Notes

The superuser can run all forms of this command.

Any user can run this command with the following options:

- `-?` (help) option
- `-V` (version) option

To run this command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`check`	`solaris.cluster.read`
`clear`	`solaris.cluster.modify`
`combine`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`

Subcommand	RBAC Authorization
monitor	solaris.cluster.modify
populate	solaris.cluster.modify
refresh	solaris.cluster.modify
rename	solaris.cluster.modify
repair	solaris.cluster.modify
replicate	solaris.cluster.modify
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.modify

Disk-path status changes are logged by using the `syslogd` command.

Each multiported tape drive or CD-ROM drive appears in the namespace once per physical connection.

Name cldevicegroup, cldg – manage Sun Cluster device groups

Synopsis

```
/usr/cluster/bin/cldevicegroup -V

/usr/cluster/bin/cldevicegroup [subcommand] -?

/usr/cluster/bin/cldevicegroup subcommand [options]
 -v [devicegroup ...]

/usr/cluster/bin/cldevicegroup add-device -d device[,...]
 devicegroup

/usr/cluster/bin/cldevicegroup add-node -n node[,...]
 [-t devicegroup-type[,...] ] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup create -n node[,...]
 -t devicegroup-type [-d device[,...] ]
 [-p name=value] devicegroup ...

/usr/cluster/bin/cldevicegroup create -i {- | clconfigfile}
 [-d device[,...] ] [-n node[,...] ] [-p name=value]
 [-t devicegroup-type[,...]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup delete [-t devicegroup-type[,...] ]
 {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup disable [-t devicegroup-type[,...] ]
 {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup enable [-t devicegroup-type[,...] ]
 {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup export [-n node[,...] ]
 [-o {- | clconfigfile}] [-t devicegroup-type[,...] ]
 {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup list [-n node[,...] ]
 [-t devicegroup-type[,...] ] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup offline [ -t devicegroup-type[,...] ]
 {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup online [-e] [-n node ]
 [-t devicegroup-type[,...] ] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup remove-device
 -d device[,...] devicegroup

/usr/cluster/bin/cldevicegroup remove-node
 -n node[,...] [-t devicegroup-type[,...] ]
 {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup set -p name=value
 [-p name=value]... [-d device[,...] ] [-n node[,...] ]
 [-t devicegroup-type[,...] ] {+ | devicegroup ...}
```

```
/usr/cluster/bin/cldevicegroup show [-n node[,...] ]
 [-t devicegroup-type[,...] ] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup status [-n node[,...] ]
 [-t devicegroup-type[,...] ] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup switch -n node [-t devicegroup-type[,...] ]
 {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup sync [-t devicegroup-type[,...] ]
 {+ | devicegroup ...}
```

Description The `cldevicegroup` command manages Sun Cluster device groups. The `cldg` command is the short form of the `cldevicegroup` command. These two commands are identical. You can use either form of the command.

The general form of this command is as follows:

`cldevicegroup` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the `-?` option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are given with the description of the option in the OPTIONS section of this man page.

With the exception of `list`, `show`, and `status`, most subcommands require at least one operand. Many subcommands accept the plus sign (`+`) as an operand to indicate `all` applicable objects. Refer to the SYNOPSIS and other sections of this man page for details.

Each subcommand can be used for all device-group types, except for the following subcommands:

- The `add-device` and `remove-device` subcommands are only valid for the `rawdisk` type.
- The `add-node`, `create`, `delete`, and `remove-node` subcommands are only valid for the `rawdisk` and `vxvm` types.

You can use some forms of this command in a non-global zone, referred to simply as a zone. For more information about valid uses of this command in zones, see the descriptions of the individual subcommands. For ease of administration, use this command in the global zone.

Subcommands The following subcommands are supported:

`add-device`
Adds new member disk devices to an existing raw-disk device group.

You can use this subcommand only in the global zone.

You can only use the `add-device` subcommand on existing device groups of the type `rawdisk`. For more information about device-group types, see the description of the `-t` option.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about how to remove disk devices from a raw-disk device group, see the description of the `remove-device` subcommand.

`add-node`
Adds new nodes to an existing device group.

You can use this subcommand only in the global zone.

This subcommand supports only the `rawdisk` and `vxvm` device-group types. You cannot add a node to an `svm` or `sds` device group by using Sun Cluster commands. Instead, use Solaris Volume Manager commands to add nodes to Solaris Volume Manager disk sets. Disk sets are automatically registered with Sun Cluster software as `svm` or `sds` device groups. For more information about device-group types, see the description of the `-t` option.

You cannot use this subcommand on a device group if the `preferenced` property for the device group is set to `true`.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about how to remove nodes from a device group, see the description of the `remove-node` subcommand.

`create`
Creates a new device group.

You can use this subcommand only in the global zone.

This subcommand supports only the `rawdisk` and `vxvm` device-group types. You cannot create an `svm` or `sds` device group by using Sun Cluster commands. Instead, use Solaris Volume Manager commands to create Solaris Volume Manager disk sets. Disk sets are automatically registered with Sun Cluster software as `svm` or `sds` device groups. For more information about device-group types, see the description of the `-t` option.

If you specify a configuration file with the `-i` option, you can supply a plus sign (+) as the operand. When you use this operand, the command creates all device groups that are specified in the configuration file that do not already exist.

For device groups of type `rawdisk`, use the `-d` option with the `create` subcommand to specify one or more devices to the device group. When you specify devices, use one `-d` option per command invocation. You cannot create multiple raw-disk device groups in one command invocation unless you use the `-i` option.

For device groups of type `vxvm`, use the `-p localonly={true|false}` option with the `create` subcommand to change a localonly VxVM disk group to or from a VxVM device group.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about how to delete device groups, see the description of the `delete` subcommand.

`delete`
Deletes device groups.

You can use this subcommand only in the global zone.

This subcommand supports only the `rawdisk` and `vxvm` device-group types.

You cannot delete `svm` or `sds` device groups by using Sun Cluster commands. To delete `svm` or `sds` device groups, instead use Solaris Volume Manager commands to delete the underlying Solaris Volume Manager disk sets.

Device groups must be offline before you can delete them.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about how to create device groups, see the description of the `create` subcommand.

`disable`
Disables offline device groups.

You can use this subcommand only in the global zone.

The disabled state of device groups survives reboots.

Before you can take an enabled device group offline, you must first clear the enabled state of the device group by using the `disable` subcommand.

If a device group is currently online, the `disable` action fails and does not disable the specified device groups.

You cannot bring a disabled device group online by using the `switch` subcommand or the `online` subcommand. You must first use the `enable` subcommand to clear the disabled state of the device group.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about how to enable device groups, see the description of the `enable` subcommand.

`enable`
Enables device groups.

You can use this subcommand only in the global zone.

The disabled state of device groups survives reboots.

Before you can bring a disabled device group online, you must first clear the disabled state of the device group by using the `enable` subcommand.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about how to disable device groups, see the description of the `disable` subcommand.

`export`
Exports the device-group configuration information.

You can use this subcommand only in the global zone.

If you specify a file name with the `-o` option, the configuration information is written to that new file. If you do not supply the `-o` option, the output is written to standard output.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list`
Displays a list of device groups.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

By default, this subcommand lists all device groups in the cluster for which the `autogen` property is set to `false`. To display all device groups in the cluster, also specify the `-v` option.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`offline`
Takes device groups offline.

You can use this subcommand only in the global zone.

If a device group is enabled, you must disable it by running the `disable` subcommand before you run the `offline` subcommand.

To start an offline device group, you can perform any of the following actions:

- Issue an explicit `online` subcommand or `switch` subcommand.
- Access a device within the device group.
- Mount a file system that depends on the device group.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

For information about how to bring device groups online, see the description of the `online` subcommand.

`online`
Brings device groups online on a pre-designated node.

You can use this subcommand only in the global zone.

If a device group is disabled, you must enable it in one of the following ways before you can bring the device group online:

- Use the `-e` option with the `online` subcommand.
- Run the `enable` subcommand before you run the `online` subcommand.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

For information about how to take device groups offline, see the description of the `offline` subcommand.

`remove-device`
Removes member disk devices from a raw-disk device group.

You can use this subcommand only in the global zone.

The `remove-device` subcommand is only valid with device groups of type `rawdisk`. This subcommand is not valid with `svm`, `sds`, and `vxvm` device-group types.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about how to add disk devices to a raw-disk device groups, see the description of the `add-device` subcommand.

`remove-node`
Removes nodes from existing device groups.

You can use this subcommand only in the global zone.

This subcommand supports only the `rawdisk` and `vxvm` device-group types. You cannot remove a node from an `svm` or `sds` device group by using Sun Cluster commands. Instead, use Solaris Volume Manager commands to remove nodes from Solaris Volume Manager disk sets. Disk sets are automatically registered with Sun Cluster software as `svm` or `sds` device groups. For more information about device-group types, see the description of the `-t` option.

You cannot use the `remove-node` subcommand on a device group if the `preferenced` property is set to `true`.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about how to add nodes to a device group, see the description of the `add-node` subcommand.

`set`
Modifies attributes that are associated with a device group.

You can use this subcommand only in the global zone.

For device groups of type `rawdisk`, use the `-d` option with the `set` subcommand to specify a new list of member disk devices for the specified device group.

If you specify the `+` operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`show`
Generates a configuration report for device groups.

You can use this subcommand only in the global zone.

By default, this subcommand reports on all device groups in the cluster for which the `autogen` property is set to `false`. To display all device groups in the cluster, also specify the `-v` option.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`status`
Generates a status report for device groups.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

By default, this subcommand reports on all device groups in the cluster for which the `autogen` property is set to `false`. To display all device groups in the cluster, also specify the `-v` option.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`switch`
Transfers device groups from one primary node in a Sun Cluster configuration to another node.

You can use this subcommand only in the global zone.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`sync`
Synchronizes device-group information with the clustering software.

You can use this subcommand only in the global zone.

Use this subcommand whenever you add or remove a VxVM volume from a VxVM device group or change any volume attribute, such as owner, group, or access permissions.

Also use the `sync` subcommand to change a device-group configuration to a replicated or non-replicated configuration.

For device groups that contain disks that use Hitachi TrueCopy data replication, this subcommand synchronizes the device-group configuration and the replication configuration. This synchronization makes Sun Cluster software aware of disks that are configured for data replication and enables the software to handle failover or switchover as necessary.

After you create a Solaris Volume Manager disk set that contain disks that are configured for replication, you must run the `sync` subcommand for the corresponding `svm` or `sds` device group. A Solaris Volume Manager disk set is automatically registered with Sun Cluster software as an `svm` or `sds` device group, but replication information is not synchronized at that time.

For newly created `vxvm` and `rawdisk` device-group types, you do not need to manually synchronize replication information for the disks. When you register a VxVM disk group or a raw-disk device group with Sun Cluster software, the software automatically discovers any replication information on the disks.

If you specify the + operand, only device groups that have the `autogen` property set to `false` are affected. To apply the command to device groups that are automatically created by the system at boot time, which have the `autogen` property set to `true`, you must explicitly specify each device group.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

Options The following options are supported:

`-?`
`--help`
Displays help information.

You can use this option either alone or with a subcommand.

- If you use this option alone, the list of available subcommands is printed.
- If you use this option with a subcommand, the usage options for that subcommand are printed.

When you use this option, no other processing is performed.

`-d` *device*[,...]
`--device=`*device*[,...]
`--device` *device*[,...]
Specifies the list of disk devices to be members of the specified raw-disk device group.

The `-d` option is only valid with the `create` and `set` subcommands for device groups of type `rawdisk`. You must always supply the entire node list. You cannot use this option to add or remove individual disks from the member disk list.

Specify disks only by the DID global device name, for example, `d3`. See the did(7) man page for more information.

`-e`
`--enable`
Enables a device group. This option is only valid when used with the `online` subcommand.

If the specified device group is already enabled, the `-e` option is ignored and the command proceeds to bring the device group online.

`-i` {- | *clconfigfile*}
`--input=`{- | *clconfigfile*}
`--input` {- | *clconfigfile*}
Specifies configuration information that is to be used for creating device groups. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through standard input. To specify standard input, supply the minus sign (`-`) instead of a file name.

The `-i` option affects only those device groups that you include in the fully qualified device-group list.

Options that you specify in the command override any options that are set in the configuration file. If configuration parameters are missing in the cluster configuration file, you must specify these parameters on the command line.

`-n` *node*[,...]
`--node=`*node*[,...]
`--node` *node*[,...]
Specifies a node or a list of nodes.

By default, the order of the node list indicates the preferred order in which nodes should attempt to take over as the primary node for a device group. The exception is for local-only disk groups which are outside Sun Cluster control and therefore the concept of primary and secondary nodes does not apply.

If the `preferenced` property of the device group is set to `false`, the order of the node list is ignored. Instead, the first node to access a device in the group automatically becomes the

primary node for that group. See the `-p` option for more information about setting the `preferenced` property for a device-group node list.

You cannot use the `-n` option to specify the node list of an `svm` or `sds` device group. You must instead use Solaris Volume Manager commands or utilities to specify the node list of the underlying disk set.

The `create` and `set` subcommands use the `-n` option to specify a list of potential primary nodes only for a device group of type `rawdisk` and `vxvm`. You must specify the entire node list of the device group. You cannot use the `-n` option to add or remove an individual node from a node list.

The `switch` subcommand uses the `-n` option to specify a single node as the new device-group primary.

The `export`, `list`, `show`, and `status` subcommands use the `-n` option to exclude from the output those device groups that are not online on the specified nodes.

The concept of primary and secondary nodes does not apply to localonly disk groups, which are outside the control of Sun Cluster.

`-o` {`-` | *clconfigfile*}
`--output`=`{-` | *clconfigfile*}
`--output` {`-` | *clconfigfile*}
Displays the device-group configuration in the format that is described by the clconfiguration(5CL) man page. This information can be written to a file or to standard output.

If you supply a file name as the argument to this option, the command creates a new file and the configuration is printed to that file. If a file of the same name already exists, the command exits with an error. No change is made to the existing file.

If you supply the minus sign (`-`) as the argument to this option, the command displays the configuration information to standard output. All other standard output for the command is suppressed.

The `-o` option is only valid with the `export` subcommand.

`-p` *name*=*value*
`--property`=*name*=*value*
`--property` *name*=*value*
Sets the values of device-group properties.

The `-p` option is only valid with the `create` and `set` subcommands. Multiple instances of `-p` *name*`-=`*value* are allowed.

The following properties are supported:

`autogen`
The `autogen` property can have a value of `true` or `false`. The default is `false` for manually created device groups. For system-created device groups, the default is `true`.

The `autogen` property is an indicator for the `list`, `show`, and `status` subcommands. These subcommands do not list devices that have the `autogen` property set to `true` unless you use the `-v` option.

This property is valid only for device groups of type `rawdisk`. See the `-t` option for more information about device-group types.

`failback`
The `failback` property can have a value of `true` or `false`. The default is `false`.

The `failback` property specifies the behavior of the system if a device-group primary node leaves the cluster membership and later returns.

When the primary node of a device group leaves the cluster membership, the device group fails over to the secondary node. When the failed node rejoins the cluster membership, the device group can either continue to be mastered by the secondary node or fail back to the original primary node.

- If the `failback` property is set to `true`, the device group becomes mastered by the original primary node.
- If the `failback` property is set to `false`, the device group continues to be mastered by the secondary node.

By default, the `failback` property is disabled during device group creation. The `failback` property is not altered during a `set` operation.

`localonly`
The `localonly` property can have a value of `true` or `false`. The default is `false`.

The `localonly` property is only valid for disk groups of type `rawdisk` and `vxvm`.

If you want a disk group to be mastered only by a particular node, configure the disk group with the property setting `localonly=true`. A local-only disk group is outside the control of Sun Cluster software. You can specify only one node in the node list of a local-only disk group. When you set the `localonly` property for a disk group to `true`, the node list for the disk group must contain only one node.

`numsecondaries`
The `numsecondaries` property must have an integer value greater than `0` but less than the total number of nodes in the node list. The default is `1`.

This property setting can be used to dynamically change the desired number of secondary nodes for a device group. A secondary node of a device group can take over as the primary node if the current primary node fails.

You can use the `numsecondaries` property to change the number of secondary nodes for a device group while maintaining a given level of availability. If you remove a node from the secondary-nodes list of a device group, that node can no longer take over as a primary node.

The `numsecondaries` property only applies to the nodes in a device group that are currently in cluster mode . The nodes must also be capable of being used together with the device group's `preferenced` property. If a group's `preferenced` property is set to `true`, the nodes that are least preferred are removed from the secondary-nodes list first. If no node in a device group is flagged as preferred, the cluster randomly picks the node to remove.

When a device group's actual number of secondary nodes drops to less that the desired level, each eligible node that was removed from the secondary-nodes list is added back to the list. Each node must meet all of the following conditions to be eligible to add back to the secondary-nodes list:

- The node is currently in the cluster.
- The node belongs to the device group
- The node is not currently a primary node or a secondary node.

The conversion starts with the node in the device group that has the highest preference. More nodes are converted in order of preference until the number of desired secondary nodes is matched.

If a node joins the cluster and has a higher preference in the device group than an existing secondary node, the node with the lesser preference is removed from the secondary-nodes list. The removed node is replaced by the newly added node. This replacement only occurs when more actual secondary nodes exist in the cluster than the desired level.

See the `preferenced` property for more information about setting the `preferenced` property for a device-group node list.

`preferenced`

The `preferenced` property can have a value of `true` or `false`. The default is `true`.

During the creation of a device group, if the `preferenced` property is set to `true`, the node list also indicates the preferred-node order. The preferred-node order determines the order in which each node attempts to take over as the primary node for a device group.

During the creation of a device group, if this property is set to `false`, the first node to access a device in the group automatically becomes the primary node. The order of nodes in the specified node list is not meaningful. Setting this property back to `true` without also respecifying the node list does not reactivate node ordering.

The preferred-node order is not changed during a `set` operation unless both specify the `preferenced=true` property and use the `-n` option to supply the entire node list for the device group, in the preferred order.

`-t` *devicegroup-type*[,…]
`--type=`*devicegroup-type*[,…]
`--type` *devicegroup-type*[,…]
Specifies a device-group type or a list of device-group types.

For the `create` subcommand, you can specify only one device-group type. The device group is then created for the type that you specify with this option.

For all other subcommands that accept the `-t` option, the device-group list that you supply to the command is qualified by this option to include only device groups of the specified type.

Not all subcommands and options are valid for all device-group types. For example, the `create` subcommand is valid only for the `rawdisk` and `vxvm` device-group types, but not for the `svm` or `sds` device-group type.

The `-t` option supports the following device-group types:

`rawdisk`
Specifies a raw-disk device group.

A raw disk is a disk that is not part of a volume-manager volume or metadevice. Raw-disk device groups enable you to define a set of disks within a device group. By default, at system boot a raw-disk device group is created for every device ID pseudo driver (DID) device in the configuration. By convention, the raw-disk device group names are assigned at initialization. These names are derived from the DID device names. For every node that you add to a raw-disk device group, the `cldevicegroup` command verifies that every device in the group is physically ported to the node.

The `create` subcommand creates a raw-disk device group and adds multiple disk devices to the device group. Before you can create a new raw-disk device group, you must remove each device that you want to add to the new group from the device group that was created for the device at boot time. Then you can create a new raw-disk device group that contains these devices. You specify the list of these devices with the `-d` option as well as specify the potential-primary node-preference list with the `-n` option.

To master a device group on a single specified node, use the `-p` option to configure the device group with the property setting `localonly=true`. You can specify only one node in the node list when you create a local-only device group.

The `delete` subcommand removes the device-group name from the cluster device-group configuration.

The `set` subcommand makes the following changes to a raw-disk device group:

- Changes the preference order of the potential primary node
- Specifies a new node list
- Enables or disables failback
- Sets the desired number of secondaries
- Adds more global devices to the device group

If a raw-disk device name is registered in a raw-disk device group, you cannot also register the raw-disk device name in a Solaris Volume Manager device group or in a VxVM device group.

The `sync` subcommand synchronizes a new replication configuration with the raw-disk device-group configuration that contains disks that use Hitachi TrueCopy data replication. Use the `sync` subcommand after you modify the replication configuration. All disks in a disk group should be either replicated or nonreplicated, but not a mixture of both.

`sds`
Specifies a device group that was originally created with Solstice DiskSuite™ software. With the exception of multi-owner disk sets, this device-group type is equivalent to the Solaris Volume Manager device-group type, `svm`. See the description of the `svm` device-group type for more information.

`svm`
Specifies a Solaris Volume Manager device group.

A Solaris Volume Manager device group is defined by the following components:

- A name
- The nodes upon which the group can be accessed
- A global list of devices in the disk set
- A set of properties that control actions such as potential primary preference and failback behavior

Solaris Volume Manager has the concept of a multihosted or shared disk set. A shared disk set is a grouping of two or more hosts and disk drives. The disk drives are accessible by all hosts and have the same device names on all hosts. This identical-device-naming requirement is achieved by using the raw-disk devices to form the disk set. The device ID pseudo driver (DID) allows multihosted disks to have consistent names across the cluster. Only hosts that are already configured as part of a disk set can be configured into the node list of a Solaris Volume Manager device group. When you add drives to a shared disk set, the drives must not belong to any other shared disk set.

The Solaris Volume Manager `metaset` command creates the disk set and automatically registers the disk set with Sun Cluster software as a Solaris Volume Manager device group. After you create the device group, you must use the `set` subcommand of the `cldevicegroup` command to set the node preference list and the `preferenced`, `failback`, and `numsecondaries` properties.

You can assign only one Solaris Volume Manager disk set to a device group. The device-group name must always match the name of the disk set.

You cannot use the `add-node` or `remove-node` subcommands to add or remove nodes in a Solaris Volume Manager device group. Instead, use the Solaris Volume Manager `metaset` command to add or remove nodes in the underlying Solaris Volume Manager disk set.

You cannot use the `delete` subcommand to remove a Solaris Volume Manager device group from the cluster configuration. Instead, use the Solaris Volume Manager `metaset` command to remove the underlying Solaris Volume Manager disk set.

Only the `export`, `list`, `show`, `status`, and `sync` subcommands work on Solaris Volume Manager multi-owner disk sets. You must use Solaris Volume Manager commands or utilities to create and delete the underlying disk set of a Solaris Volume Manager device group.

The `sync` subcommand synchronizes a new replication configuration with the device-group configuration that contains disks that use Hitachi TrueCopy data replication. Use the `sync` subcommand after you modify the replication configuration. All disks in a disk set should be either replicated or nonreplicated, but not a mixture of both.

`vxvm`

Specifies a VERITAS Volume Manager (VxVM) device group.

The `create` subcommand adds a new VxVM disk group to the Sun Cluster device-groups configuration. This task involves defining a name for the new device group, specifying the nodes on which this device group can be accessed, and specifying a set of properties that are used to control actions.

To access a disk group from a single specified node, use the `-p` option to configure the disk group with the property setting `localonly=true`. You can specify only one node in the node list when you create a local-only `vxvm` disk group.

You can assign only one VxVM disk group to a device group. The device group name always matches the name of the VxVM disk group. You cannot create a VxVM device group unless you first import the corresponding VxVM disk group on one of the nodes in that device group's node list.

Before you can add a node to a VxVM device group, every physical disk in the disk group must be physically ported to that node.

The `delete` subcommand removes a VxVM device group from the Sun Cluster device-groups configuration.

The `create` subcommand makes the following changes to a `vxvm` device group:

- Changes the order of the potential primary node preference
- Enables or disables failback
- Changes the desired number of secondaries
- Changes a local-only VxVM disk group to a regular VxVM disk group

The `sync` subcommand synchronizes the clustering software with VxVM disk-group volume information. Use this subcommand whenever you add or remove a volume from a device group or whenever you change any volume attribute, such as owner, group, or access permissions.

If the disk group contains disks that use Hitachi TrueCopy data replication and you modify the replication configuration, use the `sync` subcommand to synchronize the new replication configuration with the device-group configuration. All disks in a disk group should be either replicated or nonreplicated, but not a mixture of both.

You cannot use the `cldevicegroup` command on VxVM shared-disk groups.

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The `-V` option only displays the version of the command. No other operations are performed.

`-v`
`--verbose`
Displays verbose messages to standard output.

You can use this option with any form of the command.

Operands The following operand is supported:

devicegroup Specifies a device group.

The `cldevicegroup` command accepts only Sun Cluster device-group names as operands. For most forms of the command that accept more than one device-group name, you can use the plus sign (+) to specify all possible device groups.

Note – The + operand includes only manually created device groups, but ignores all automatically created device groups, which have the `autogen` property set to `true`. Sun Cluster software automatically creates such device groups at each system boot. To apply a command these "hidden" device groups, you must specify each device group explicitly.

Exit Status The complete set of exit status codes for all commands in this command set are listed on the Intro(1CL) man page.

If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

`1` `CL_ENOMEM`
Not enough swap space

`3` `CL_EINVAL`
Invalid argument

`6` `CL_EACCESS`
Permission denied

`35` `CL_EIO`
I/O error

`36` `CL_ENOENT`
No such object

`39` `CL_EEXIST`
Object exists

Examples

EXAMPLE 1 Creating a VxVM Device Group

The following example shows how to create the VxVM device group `vxdevgrp1` with the desired node list `phys-schost-1,phys-schost-2,phys-schost-3` and set the `failback` property to `true`.

```
# cldevicegroup create -t vxvm -n phys-schost-1,phys-schost-2,phys-schost-3 \\
-p failback=true vxdevgrp1
```

EXAMPLE 2 Modifying a Device Group

The following example shows how to set the `preference` property of device group `devgrp1` to `true` and set the `numsecondaries` property to 2. The command also specifies the desired node list, `phys-schost-1,phys-schost-2,phys-schost-3`.

```
# cldevicegroup set -p preferenced=true -p numsecondaries=2 \\
-n phys-schost-1,phys-schost-2,phys-schost-3 devgrp1
```

EXAMPLE 3 Modifying a Raw-Disk Device Group

The following example shows how to modify the existing raw-disk device group `rawdevgrp1`. The command specifies devices `d3` and `d4` in a new-member device list and sets the `localonly` attribute to `true`. The node `phys-schost-1` is the only primary node that is allowed for the local-only raw-disk device group.

```
# cldevicegroup set -d d3,d4 -p localonly=true -n phys-schost-1 rawdevgrp1
```

EXAMPLE 4 Resetting the `numsecondaries` Attribute of a Device Group

The following example shows how to reset the `numsecondaries` attribute of device group `devgrp1` to the appropriate system default value by specifying no value for that attribute.

```
# cldevicegroup set -p numsecondaries= devgrp1
```

EXAMPLE 5 Switching Over a Device Group

The following example shows how to switch over the device group `devgrp1` to a new master node, `phys-schost-2`.

```
# cldevicegroup switch -n phys-schost-2 devgrp1
```

EXAMPLE 6 Disabling a Device Group

The following example shows how to disable the device group `devgrp1`.

```
# cldevicegroup disable devgrp1
```

EXAMPLE 7 Taking Offline a Device Group

The following example shows how to disable and take offline the device group `devgrp1`.

```
# cldevicegroup disable devgrp1
# cldevicegroup offline devgrp1
```

EXAMPLE 8 Bringing a Device Group Online on its Primary Node

The following example shows how to bring online the device group `devgrp1` on its default primary node. The command first enables the device group.

```
# cldevicegroup online -e devgrp1
```

EXAMPLE 9 Bringing a Device Group Online on a Specified Node

The following example shows how to bring online the device group `devgrp1` on `phys-schost-2` as its new primary node.

```
# cldevicegroup switch -n phys-schost-2 devgrp1
```

EXAMPLE 10 Adding New Nodes to a Device Group

The following example shows how to add a new node, `phys-schost-3`, to the device group `devgrp1`. This device group is not of the device-group type `svm`.

```
# cldevicegroup add-node -n phys-schost-3 devgrp1
```

EXAMPLE 11 Deleting a Device Group

The following example shows how to delete the device group `devgrp1` from the Sun Cluster configuration. This device group is not of the device-group type `svm`.

```
# cldevicegroup delete devgrp1
```

EXAMPLE 12 Synchronizing Replication Information With the Device-Group Configuration

The following example shows how to make Sun Cluster software aware of the replication configuration that is used by the disks in the device group `devgrp1`.

```
# cldevicegroup sync devgrp1
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cldevice(1CL), cluster(1CL), `metaset`(1M), clconfiguration(5CL), `rbac`(5), did(7)

Sun Cluster System Administration Guide for Solaris OS

Notes

The superuser can run any forms of this command.

Any user can also run this command with the following options:

- `-?` (help) option
- `-V` (version) option

To run this command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
add-device	solaris.cluster.modify
add-node	solaris.cluster.modify
create	solaris.cluster.modify
delete	solaris.cluster.modify
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
offline	solaris.cluster.admin
online	solaris.cluster.admin
remove-device	solaris.cluster.modify
remove-node	solaris.cluster.modify
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
switch	solaris.cluster.modify
sync	solaris.cluster.admin

Name clinterconnect, clintr – manage the Sun Cluster interconnect

Synopsis

```
/usr/cluster/bin/clinterconnect  -V

/usr/cluster/bin/clinterconnect  [subcommand] -?

/usr/cluster/bin/clinterconnect  subcommand  [options]
 -v  [endpoint[,endpoint] ...]

/usr/cluster/bin/clinterconnect add  [-d] endpoint[,endpoint] ...

/usr/cluster/bin/clinterconnect add  -i {- |  clconfigfile}
 [-d] [-n node[,...] ] {+ |  endpoint[,endpoint] ...}

/usr/cluster/bin/clinterconnect disable  [-n node[,...] ]
 {+ |  endpoint[,endpoint] ...}

/usr/cluster/bin/clinterconnect enable  [-n node[,...] ]
 {+ |  endpoint[,endpoint] ...}

/usr/cluster/bin/clinterconnect export  [-o  {-
 |  configfile}] [-n node[,...] ] [+ |  endpoint[,endpoint] ...]

/usr/cluster/bin/clinterconnect remove  [-l] endpoint[,endpoint] ...

/usr/cluster/bin/clinterconnect show  [-n node[,...] ]
 [+ |  endpoint[,endpoint] ...]

/usr/cluster/bin/clinterconnect status  [-n node[,...] ]
 [+ |  endpoint[,endpoint] ...]
```

Description The `clinterconnect` command manages configuration of the cluster interconnect and displays configuration and status information. The `clintr` command is the short form of the `clinterconnect` command. The `clinterconnect` command and the `clintr` command are identical. You can use either form of the command.

The cluster interconnect consists of two endpoints which are connected with cables. An endpoint can be an adapter on a node or a switch, also called a junction. A cable can connect an adapter and a switch or connect two adapters in certain topologies. The cluster topology manager uses available cables to build end-to-end interconnect paths between nodes. The names of cluster interconnect components that are supplied to this command should accurately reflect the actual physical configuration. Failure to do so will prevent the system from building end-to-end cluster interconnect paths. This lack of functional cluster interconnects would result in cluster nodes that are unable to communicate with each other, nodes that panic, and similar conditions.

You must run the `clinterconnect` command from a cluster node that is online and is in cluster mode.

The general form of this command is as follows:

`clinterconnect` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the `-?` option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are given with the description of the option in the OPTIONS section of this man page.

You can use some forms of this command in a non-global zone, referred to simply as a zone. For more information about valid uses of this command in zones, see the descriptions of the individual subcommands. For ease of administration, use this command in the global zone.

Subcommands The following subcommands are supported:

`add`
Adds the new cluster interconnect components that are specified as operands to the command.

You can use this subcommand only in the global zone.

The syntax of the operand determines whether you are adding a cable, a switch, or an adapter. Refer to the OPERANDS section of this man page for more information.

Use the `add` subcommand to configure an interconnect cable between an adapter and either an adapter on another node or an interconnect switch. The adapter or switch endpoints that constitute the cable do not need to already exist. You can also use this subcommand to add adapters or switches to the configuration.

When you add an adapter or a switch to the configuration, the command also enables the adapter or switch. When you add a cable, the command also enables each of the cable's endpoints, if the endpoints are not already enabled.

In a two-node cluster, if you add a cable with an adapter at each endpoint, a virtual switch is also created.

Use the `-d` option to add an endpoint in the disabled state.

If you specify a configuration file with the `-i` option, you can specify the plus sign (+) as the operand. When you use this operand, the command creates all interconnect components that are specified in the configuration file which do not already exist in the cluster.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about removing interconnect components, see the description of the `remove` command.

`disable`
Disables the interconnect components that are specified as operands to the command.

You can use this subcommand only in the global zone.

The syntax of the operand determines whether you are disabling a cable, a switch, or an adapter. Refer to the OPERANDS section of this man page for more information.

If you attempt to disable an adapter or a switch that is connected to an enabled cable, the operation results in an error. You must first disable the cable before you attempt to disable the connected adapter or switch.

When you disable a cable, the command also disables each endpoint that is associated with the cable, which can be an adapter or a switch port. The command also disables the switch if all of the switch ports are in a disabled state.

If you attempt to disable the cable or an endpoint of the last cluster interconnect path of an active cluster node, the operation results in an error.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about enabling interconnect components, see the description of the `enable` subcommand.

`enable`
Enables the interconnect components that are specified as operands to the command.

You can use this subcommand only in the global zone.

The syntax of the operand determines whether you are enabling a cable, a switch, or an adapter. Refer to the OPERANDS section of this man page for more information.

When you enable a cable, the command also enables each endpoint that is associated with the cable, which can be an adapter or a switch port.

For information about disabling interconnect components, see the description of the `disable` subcommand.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`export`
Exports the cluster interconnect configuration information.

You can use this subcommand only in the global zone.

If you supply a file name with the `-o` option, the configuration information is written to that new file. If you do not use the `-o` option, the output is written to standard output.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`remove`
Removes the cluster interconnect components that are specified as operands to the command.

You can use this subcommand only in the global zone.

The syntax of the operand determines whether you are removing a cable, a switch, or an adapter. Refer to the OPERANDS section of this man page for more information.

The following behaviors apply when you remove a cable:

- You must first disable a cable before you can remove the cable.
- If you attempt to remove a cable that is enabled, the remove operation results in an error.
- If you remove a disabled cable, the cable's endpoints are also removed except in the following circumstances:
  - The switch is in use by another cable.
  - You also specify the `-l` option.

The following behaviors apply when you remove an adapter or switch endpoint:

- If you remove an endpoint that is not associated with a cable, the specified endpoint is removed.
- If you attempt to remove an endpoint that is associated with a cable, the remove operation results in an error. This occurs regardless of whether the cable is enabled or disabled.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

For information about adding interconnect components, see the description of the `add` subcommand.

`show`
Displays the configuration of the interconnect components that are specified as operands to the command.

You can use this subcommand only in the global zone.

The configuration information includes whether the component is enabled or disabled. By default, the configuration of all interconnect components is printed.

The `show` subcommand accepts the plus sign (+) as an operand to specify all components.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`status`
Displays the status of the interconnect paths. By default, the report displays the status of all interconnect paths in the system.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

The following are the possible states of an interconnect path.

`faulted`	The interconnect path has encountered an error that prevents it from functioning.
`Path online`	The interconnect path is online and is providing service.
`waiting`	The interconnect path is in transition to the `Path online` state.

To determine whether an interconnect component is enabled or disabled, use the `show` subcommand.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

Options The following options are supported:

`-?`
`--help`
Displays help information. When this option is used, no other processing is performed.

You can use this option either alone or with a subcommand.

- If you specify this option alone, the list of available subcommands is printed.
- If you specify this option with a subcommand, the usage options for that subcommand are printed.

`-d`
Specifies that the endpoint is added in the disabled state.

`-i` {`-` | *clconfigfile*}
`--input=`{`-` | *clconfigfile*`-`}
`--input` {`-` | *clconfigfile*`-`}
Specifies configuration information that is to be used for adding or modifying cables. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through standard input. To specify standard input, supply the minus sign (`-`) instead of a file name.

Options that you specify in the command override any options that are set in the cluster configuration file. If required elements are missing from a cluster configuration file, you must specify these elements on the command line.

You can use the minus sign (`-`) argument with this option to specify that the configuration is supplied as standard input.

`-l`
`--limited`
Specifies that the cable removal operation removes only the cable but not any of its endpoints.

The `-l` option is only valid with the `remove` subcommand. If you do not specify this option with the `remove` subcommand, the command removes the specified cables as well as any associated adapters. In addition, if the cable removal operation removes the last connection to a switch, the command also removes the switch from the configuration.

`-n` *node*[,…]
`--node=`*node*[,…]
`--node` *node*[,…]
Specifies a node or list of nodes. Use this option to limit the operation to adapters and cables that are attached only to the specified node.

You can specify a node either by its node name or by its node ID.

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile*}
Displays the interconnect configuration in the format that is described by the clconfiguration(5CL) man page.

Only the `export` subcommand accepts the `-o` option.

If you supply a file name as the argument to this option, the command creates a new file and the configuration is printed to that file. If a file of the same name already exists, the command exits with an error. No change is made to the existing file.

If you supply the minus sign (`-`) as the argument to this option, the command displays the configuration information to standard output. All other standard output for the command is suppressed.

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The `-V` option only displays the version of the command. No other operations are performed.

`-v`
`--verbose`
Displays verbose messages to standard output.

You can use this option with any form of the command.

Operands This command accepts interconnect endpoints or pairs of comma-separated endpoints as operands. An endpoint can be an adapter or a switch. A comma-separated pair of endpoints indicates a cable.

For those forms of the command that accept more than one interconnect component, you can use the plus sign (+) argument to specify all possible components.

The following operands are supported:

node:*adapter*
Specifies an adapter endpoint.

An adapter endpoint has a node name and an adapter name. The adapter name is constructed from an interconnect name that is immediately followed by a physical-unit number, such as `hme0`. The node that hosts the adapter does not need to be active in the cluster for these operations to succeed.

The following types of adapters can be configured as cluster transport adapters:

Ethernet	You can connect an Ethernet adapter to another Ethernet adapter or to an Ethernet switch.
InfiniBand	You can connect an InfiniBand adapter only to an InfiniBand switch.
SCI-PCI	You can connect an SCI adapter only to another SCI adapter or to an SCI switch. The form of the SCI adapter name is `sci`*N*, where *N* is the physical-unit number.

By default, adapters are configured as using the `dlpi` transport type.

To specify a tagged-VLAN adapter, use the tagged-VLAN adapter name that is derived from the physical device name and the VLAN instance number. The VLAN instance number is the VLAN ID multiplied by 1000 plus the original physical-unit number. For example, a VLAN ID of `11` on the physical device `ce2` translates to the tagged-VLAN adapter name `ce11002`.

switch[*@port*]
Specifies a switch endpoint.

Each interconnect switch name must be unique across the namespace of the cluster. You can use letters, digits, or a combination of both. The first character of the switch name must be a letter.

If you do not supply a *port* component for a switch endpoint, the command assumes the default port name. The default port name is equal to the node ID of the node that is attached to the other end of the cable.

You can configure the following types of switches as cluster transport switches:

Dolphin SCI	Use the Dolphin SCI switch with SCI-PCI adapters. When you connect an SCI-PCI adapter to a port on an SCI switch, the port name must match the port number that is printed on the SCI-PCI switch. Failure to supply the correct port name results in the same condition as when the cable between an adapter and a switch is removed.

Ethernet	Use the Ethernet switch with Ethernet adapters.
InfiniBand	Use the InfiniBand switch with InfiniBand adapters.

By default, switches are configured as using the `switch` type.

node:*adapter*,*node*:*adapter*
node:*adapter*,*switch*[*@port*]
Specifies a cable.

A cable is a comma-separated pair of adapter or switch endpoints. The order of endpoints is not important. Use cable operands to add a complete cluster interconnect. Because the `clinterconnect` command automatically creates both endpoints when you add a cable, you do not need to separately create adapter or switch endpoints.

Exit Status The complete set of exit status codes for all commands in this command set are listed on the Intro(1CL) man page.

If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

`1` `CL_ENOMEM`
Not enough swap space

`3` `CL_EINVAL`
Invalid argument

`6` `CL_EACCESS`
Permission denied

`35` `CL_EIO`
I/O error

`36` `CL_ENOENT`
No such object

`37``CL_EOP`
Operation not allowed

`38` `CL_EBUSY`
Object busy

`39` `CL_EEXIST`
Object exists

Examples

EXAMPLE 1 Creating a Direct-Connect Cluster Interconnect Cable

The following example shows how to add a cable that connects ports between the adapter `hme0` on the node `phys-schost-1` and the adapter `hme0` on the node `phys-schost-2`.

```
# clinterconnect add phys-schost-1:hme0,phys-schost-2:hme0
```

EXAMPLE 2 Creating a Cable Between a Switch and an Adapter

The following example shows how to add a cable between the adapter `hme0` on the node `phys-schost-1` and the switch `ether_switch`.

```
# clinterconnect add phys-schost-1:hme0,ether_switch
```

EXAMPLE 3 Disabling a Cable

The following example shows how to disable the cable that is connected between the adapter `hme0` on the node `phys-schost-1` and the switch `ether_switch`.

```
# clinterconnect disable phys-schost-1:hme0,ether_switch
```

EXAMPLE 4 Removing a Cluster Interconnect Cable

The following example shows how to remove the cable that is connected between the adapter `hme0` on the node `phys-schost-1` and the switch `ether_switch`.

```
# clinterconnect remove phys-schost-1:hme0,ether_switch
```

EXAMPLE 5 Creating a Cable Between a Tagged-VLAN Adapter and a Switch

The following example shows how to add a cable between the tagged VLAN adapter `ce73002` on the node `phys-schost-1` and the VLAN-capable switch `switch1`. The physical name of the adapter is `ce2` and the VLAN ID is `73`.

```
# clinterconnect add phys-schost-1:ce73002,switch1
```

EXAMPLE 6 Enabling a Switch

The following example shows how to enable the switch endpoint `switch1`.

```
# clinterconnect enable switch1
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also Intro(1CL), cluster(1CL), clconfiguration(5CL), `rbac`(5)

Sun Cluster 3.1 - 3.2 Hardware Administration Manual for Solaris OS

Sun Cluster Software Installation Guide for Solaris OS

Notes The superuser can run all forms of this command.

Any user can run this command with the following options.

- `-?` (help) option
- `-V` (version) option

To run this command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`add`	`solaris.cluster.modify`
`disable`	`solaris.cluster.modify`
`enable`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`remove`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`
`status`	`solaris.cluster.read`

Name clnasdevice, clnas – manage access to NAS devices for Sun Cluster

Synopsis

```
/usr/cluster/bin/clnasdevice -V

/usr/cluster/bin/clnasdevice [subcommand]
 -?

/usr/cluster/bin/clnasdevice subcommand
 [options] -v [nasdevice[ ...]]

/usr/cluster/bin/clnasdevice add  -t type {-pname=value[,...]
 |  -u userid} [-f passwdfile] nasdevice

/usr/cluster/bin/clnasdevice add  -i {- |  clconfigfile}
 [-t type] [-pname=value |  -u userid] {-f passwdfile}
 {+ |  nasdevice[ ...]}

/usr/cluster/bin/clnasdevice add-dir -d directory[,...]
 nasdevice

/usr/cluster/bin/clnasdevice add-dir -i {- |  clconfigfile}
 [-d all |  directory[,...]] [-f passwordfile]
 {+ |  nasdevice[ ...]}

/usr/cluster/bin/clnasdevice export  [-o  {- |  clconfigfile}]
 [-t type[,...]] [-d all |  directory[,...]]
 [+ |  nasdevice[ ...]]

/usr/cluster/bin/clnasdevice list [-t type[,...]]
 [+ |  nasdevice[ ...]]

/usr/cluster/bin/clnasdevice remove [-t type[,...]]
 [-F ] {+ |  nasdevice[ ...]}

/usr/cluster/bin/clnasdevice remove-dir -d all
 |  directory[,...] nasdevice

/usr/cluster/bin/clnasdevice set {-pname=value[,...]
 |  -u userid} [-f passwdfile] nasdevice

/usr/cluster/bin/clnasdevice show [-t type[,...]]
 [+ | nasdevice[ ...]]
```

Description The `clnasdevice` command manages Sun Cluster configuration information about NAS devices and their directories.

The `clnas` command is the short form of the `clnasdevice` command. The `clnas` and `clnasdevice` commands are identical. You can use either form of the command.

The general form of this command is as follows:

`clnasdevice` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the `-?`option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are provided with the description of the option in the "OPTIONS" section of this man page.

Before you use the `clnasdevice` command to configure a NAS device in the cluster, your NAS device must conform to the following conditions:

- The NAS device must be set up and operating.
- The NAS device must be booted and running.
- The NAS device's directories must be created and made available to the cluster nodes.
- If the NAS device will be a quorum device, the LUN for the quorum device must be created. For information on configuring a NAS quorum device, see the `clquorum`.1cl man page

Depending on the NAS device vendor, you might need to perform additional tasks before you configure the device into the cluster. For details about these tasks, see the `-t` option in "OPTIONS". Refer to the documentation for your particular NAS device for procedures about setting up a NAS device and exporting the directories.

After the NAS device is fully operational and ready to provide storage for the cluster, use the `clnasdevice` command to manage the NAS device configuration information in the cluster. Otherwise, the cluster cannot detect the NAS device and its exported directories. Consequently, the cluster cannot protect the integrity of the information in these directories.

Use the `clnasdevice` command for these administrative tasks:

- To create the NAS device configuration
- To update NAS type-specific properties
- To remove the NAS device's directories from the cluster configuration
- To remove the NAS device from the cluster configuration

The `clnasdevice` command can be run only on an active cluster node. The result of running the command is always the same, regardless of the node on which you run it. You can use this command only in the global zone.

Subcommands The following subcommands are supported:

`add`
Adds a NAS device to the Sun Cluster configuration.

Use the `-t` option to specify the vendor of the NAS device. For details, see the `-t` option description in the "OPTIONS" section.

Depending on the type of your NAS device, you might have to set additional properties. These required properties are also explained in the `-t` option description in the "OPTIONS" section.

Users other than the superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this command. See `rbac`(5).

See also the description of the `remove` subcommand.

`add-dir`
Adds the specified directories of an already configured NAS device to the cluster configuration. You must have created these directories on the device and made them available to the cluster before using this subcommand. For information about creating directories, see the documentation for your NAS device.

You can add NAS device directories one of two ways:

- Use the `clnasdevice add` command to configure the NAS device in the cluster. Then use the `clnasdevice add-dir` command to configure that device's directories in the cluster.
- Use the `clnasdevice add-dir -i` *configurationfile* form of the command to add the device and configure its directories in a single step. To add directories using this method, provide the password file using the `-f` option. For details on this option, see the "OPTIONS" section. For the format of the clconfiguration(5CL) file, see its man page.

Whenever you create a new directory on the NAS device and make it available to the cluster nodes, you need to use this `add-dir` subcommand to add the directories to the cluster configuration.

Users other than the superuser require `solaris.cluster.modify` RBAC authorization to use this command. See `rbac`(5).

See also the description of the `remove-dir` subcommand.

`export`
Exports the cluster NAS device configuration information. If you specify a file with the `-o` option, the configuration information is written to that file. If you do not use the `-o` option, the output is written to standard output (`stdout`).

The `export` subcommand does not modify cluster configuration information.

Users other than the superuser require `solaris.cluster.read` RBAC authorization to use this command. See `rbac`(5).

`list`
Displays the NAS devices configured in the cluster.

To display the device's directories that are configured in the cluster and the device type, use the verbose option `-v`.

To display NAS devices of a particular type, use the `-t` option.

Users other than the superuser require `solaris.cluster.read` RBAC authorization to use this command. See `rbac`(5).

`remove`
Removes the specified NAS device or devices from the Sun Cluster configuration.

If you do not specify the force option, `-F`, you must have already removed the NAS device directories from the configuration by using the `remove-dir` subcommand.

If you specify the force option, `-F`, the command removes the NAS device and its directories from the cluster configuration. See `-F` in *OPTIONS*.

Users other than the superuser require `solaris.cluster.modify` RBAC authorization to use this command. See `rbac`(5).

See also the description of the `add` subcommand.

`remove-dir`
Removes the specified NAS directory or directories from the Sun Cluster configuration.

The `remove-dir` subcommand removes the exported directories specified by the `-d` option. When you use `-d all`, the subcommand removes all the directories of the specified NAS device.

Whenever a directory is removed from the NAS device, you need to use this `remove-dir` subcommand to remove the directories from the cluster configuration. The NAS directories in the cluster configuration must match the existing directories that are exported from the NAS device.

Users other than the superuser require `solaris.cluster.modify` RBAC authorization to use this command. See `rbac`(5).

See also the description of the `add-dir` subcommand.

`set`
Sets specified properties of a specific NAS device.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

Users other than the superuser require `solaris.cluster.modify` RBAC authorization to use this command. See `rbac`(5).

`show`
When no options are provided, displays the following information:

- A listing of all the current NAS devices configured in Sun Cluster
- The available directories of each NAS device
- All properties associated with each NAS device

To display a particular type of NAS device, specify the `-t` option. To display information about a particular device, pass the NAS device's hostname as the operand to the command.

Users other than the superuser require `solaris.cluster.read` RBAC authorization to use this command. See `rbac`(5).

Options The following options are supported:

`-?`
`--help`
Displays help information. When this option is used, no other processing is performed for any other options.

You can specify this option with or without a subcommand.

If you specify this option without a subcommand, the list of subcommands for this command is displayed.

If you specify this option with a subcommand, the usage options for the subcommand are displayed.

`-d` *directory*[,…]
`--directory=`*directory*`-[,...]`
`--directory` *directory*`-[,...]`
Specifies the NAS device directory or directories. Use this option only with the `add-dir`, `remove-dir`, and `export` subcommands.

This option accepts a special keyword, `all`. When you use the `-d all` option, you specify all directories on the specified NAS devices.

- With the `remove-dir` subcommand, all directories on the specified devices are removed.
- With the `export` subcommand, the configuration information of all directories on the specified devices is displayed to the specified output.
- With the `add-dir` subcommand and the `-i` *configfile* option, all directories on the specified NAS device that are listed in the configuration file are added.

`-F`
`--force`
Forces the removal of the specified NAS device.

The force option is available only with the `remove` subcommand. When you use this force option with the `remove` subcommand, it removes the NAS device and its configured directories from the Sun Cluster configuration.

`-f` *passwdfile*
`--passwdfile=`*passwd_file*
`--passwdfile` *passwd_file*
Specifies the password file that contains the password to use when logging in to the NAS device.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

For security reasons, the password cannot be specified in command-line options. To keep the password secure, place it in a text file and specify the file by using the `-f` option. If you do not specify an input file for the password, the command prompts for the password.

Set permissions of the input file to readable by root and prohibit access by either group or world.

When using `clnasdevice add` with the `-i` option, if your *clconfigfile* does not contain the password the `-f` *passwdfile* option is required.

In the input file, observe the following restrictions:

- Specify passwords on a single line. Do not type passwords across multiple lines.
- Leading white spaces and tabs are ignored.
- Comments begin with an unquoted # sign. Comments continue to the next new line

  The parser ignores all comments.
- If you use an input file for the device user password, the # sign cannot be part of the password.

`-i` *clconfigfile*
`--input={- | clconfigfile-}`
`--input {- | clconfigfile-}`
Specifies the configuration information that is used to create or modify the NAS device. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or through the standard input (`stdin`). To specify the standard input, specify `-` instead of a file name.

If you specify the same properties on the command line and in the *clconfigfile* file, the properties that you set on the command-line prevail.

When using `clnasdevice add` with the `-i` option, the `-f` *passwdfile* option is required.

`-o` {`-` | *clconfigfile*}
`--output={- | clconfigfile-}`
`--output {- | clconfigfile-}`
Writes the NAS device configuration information in the format that is defined in the clconfiguration(5CL) man page. You can write this information to a file or to the standard output (`stdout`). To specify the standard output, specify `-` instead of a file name.

`-p` *name*=*value*[,…]
`--property=`*name*`-=`*value*`-[,...]`
`--property` *name*`-` *value*`-[,...]`
Specifies the properties that are specific to a NAS device type.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

You must specify this option when you use the `add` subcommand to add a new NAS device to a cluster configuration. You also must specify this option when you modify the properties of a NAS device with the `set` subcommand. See the description of the `-t` option for more information.

`-t` *type*
`--type=`*nas_device_type*
`--type` *nas_device_type*
Specifies the NAS device type. You must specify that this option is required when you add a NAS device to the Sun Cluster configuration. The NAS device type is identified by the vendor name. For example, the NAS device type for Sun Microsystems, Inc. is `sun`.

Different types of NAS devices have different or in some cases, no properties.

Specifies the NAS device type. You must specify this option when you add a NAS device to the Sun Cluster configuration. The NAS device type is identified by the vendor name. For example, the NAS device type for Sun Microsystems, Inc. is `sun`.

Different types of NAS devices have different or in some cases, no properties.

`sun`	Specifies a NAS device from Sun Microsystems, Inc. **Note –** You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.
`sun`	Specifies a NAS device from Sun Microsystems, Inc. **Note –** You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

`netapp`

Specifies a NAS device from Network Appliance, Inc. The NAS device from Network Appliance, Inc. has the following property. This property is required if you add a NAS device by using the `add` subcommand:

`-p` userid=*userid* [`-f` *passwdfile*]

or

`-u` *userid* [`-f` *passwdfile*]

The `userid` is the user ID that the cluster uses to perform administrative duties on the NAS device. When you add a user ID to the device configuration, you are prompted for its password. You can also place the password in a text file and use it by specifying the `-f` option.

Before adding a NAS device and its exported directories into the cluster configuration, you must have already performed the following tasks:

- Set up the NAS device.
- Set up directories and made them available to the cluster nodes.
- Determined the user ID and password to use for administrative tasks on the device.

The NAS device must also be up and running. To provide support for `netapp` NAS devices in the cluster, the administrator must also install the required software module that is provided by Network Appliance, Inc. Additionally, the iSCSI license must be valid for the Network Appliance, Inc. NAS device. For instructions about obtaining the support module, see *Sun Cluster With Network-Attached Storage Devices Manual for Solaris OS*.

`-u` *userid*
`--userid=`*userid*
`--userid` *userid*

Specifies the user ID that is used to log in to the NAS device.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `set` subcommand and the `-f`, `-p`, and `-u` options do *not* apply.

The cluster needs to know the user ID to log in and perform administrative duties on the device.

Alternatively, you can specify the user ID with the `-p` option. See `-p` for details.

You can use this option only with the `add` and `set` subcommands.

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The version of the command is displayed. No other processing is performed.

`-v`
`--verbose`
Displays verbose information to standard output (`stdout`).

Operands The following operands are supported:

nasdevice
The name of a NAS device. The NAS device name is the hostname by which the NAS device communicates over the network. The cluster needs the NAS hostname of the NAS device to communicate with the NAS device. If the subcommand accepts more than one NAS device, you can use the plus sign (+) to specify all NAS devices. For the `add` and `add-dir` subcommands, the plus sign operand indicates all NAS devices in the specified configuration file.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

The following exit values can be returned:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

15 CL_EPROP
Invalid property

The property or value that you specified with the `-p`, `-y`, or `-x` option does not exist or is not allowed.

18 CL_EINTERNAL
Internal error was encountered

An internal error indicates a software defect or other defect.

36 CL_ENOENT
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

39 CL_EEXIST
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

41 CL_ETYPE
Invalid type

The type that you specified with the `-t` or `-p` option does not exist.

Examples

EXAMPLE 1 Adding a NAS Device From Sun Microsystems, Inc. to a Cluster

The following `clnasdevice` command adds the Sun Microsystems, Inc. storage system `sunapp1` to the Sun Cluster configuration.

```
# clnasdevice add -t sun sunapp1
```

EXAMPLE 2 Adding a NAS Device From Sun Microsystems, Inc. to a Cluster

The following `clnasdevice` command adds the Sun Microsystems, Inc. storage system `sunnas1` to the Sun Cluster configuration.

```
# clnasdevice add -t sun sunnas1
```

EXAMPLE 3 Adding a NAS Device From Network Appliance, Inc. to a ClusterAdding a NAS Device From Network Appliance, Inc. to a Cluster

The following `clnasdevice` command adds the Network Appliance, Inc. storage system `netapp1` to the Sun Cluster configuration.

```
# clnasdevice add -t netapp -p userid=root netapp1
Please enter password
```

EXAMPLE 4 Adding Some NAS-Device-Directories to a Cluster

The following `clnasdevice` command adds two exported directories of the already configured NAS device `sunapp1sunnas1` to the cluster configuration.

```
# clnasdevice add-dir -d /export/dir1,/export/dir2 sunapp1sunnas1
```

EXAMPLE 5 Removing All NAS-Device-Directories From Cluster Configuration

The following `clnasdevice` command removes all directories that belong to the NAS device `netapp1` from the cluster configuration.

```
# clnasdevice remove-dir -d all netapp1
```

EXAMPLE 6 Removing a NAS Device From a Cluster

The following `clnasdevice` command removes the NAS device `sunapp1sunnas1` and all of its remaining directories, if any, from the Sun Cluster configuration.

```
# clnasdevice remove -F sunapp1sunnas1
```

EXAMPLE 7 Displaying the NAS Devices Configured in the Cluster

The following `clnasdevice` command displays the names of all NAS devices that are configured in the cluster. To see a listing of the devices and their directories, use the verbose option or the `show` subcommand.

```
# clnasdevice list
sunappsunnas1
```

EXAMPLE 8 Display the NAS Devices and Their Directories

The following `clnasdevice` command displays the names of all NAS devices that are configured in the cluster, along with their directories that are part of the cluster configuration.

```
# clnasdevice show -v
Nas Device:    sunapp1sunnas1.sfbay.sun.com
Type:          sunappsun
```

EXAMPLE 8 Display the NAS Devices and Their Directories *(Continued)*

```
Userid:         root
Directory:      /export/dir1
                /export/dir2

Nas Device:     netapp2
Type:           netapp
Userid:         root
Directory:      /export/dir1
                /export/dir2
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cluster(1CL)

Notes

The superuser can run all forms of this command.

Any user can run this command with the following subcommands and options:

- `-?` option
- `-V` option

To run this command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`add`	`solaris.cluster.modify`
`add-dir`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`set`	`solaris.cluster.modify`
`remove`	`solaris.cluster.modify`
`remove-dir`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`

Name clnode – manage Sun Cluster nodes

Synopsis

```
/usr/cluster/bin/clnode -V

/usr/cluster/bin/clnode [subcommand] -?

/usr/cluster/bin/clnode subcommand [options] -v [node ...]

/usr/cluster/bin/clnode add {-n sponsornode}
[-i {- | clconfigfile}] [-c clustername] [-G globaldevfs]
 [-e endpoint,endpoint] node

/usr/cluster/bin/clnode clear [-F] node...

/usr/cluster/bin/clnode evacuate [-T seconds] node

/usr/cluster/bin/clnode export [-o {- | clconfigfile}]
[+ | node ...]

/usr/cluster/bin/clnode list [+ | node ...]

/usr/cluster/bin/clnode remove [-n sponsornode]
[-G globaldevfs] [-F] [node ]

/usr/cluster/bin/clnode set [-p name=value] [...]
{+ | node ...}

/usr/cluster/bin/clnode show [-p name[,...]] [+ | node ...]

/usr/cluster/bin/clnode show-rev [node]

/usr/cluster/bin/clnode status [ -m] [+ | node ...]
```

Description This command does the following:

- Adds a node to the cluster
- Removes a node from the cluster
- Attempts to switch over all resource groups and device groups
- Modifies the properties of a node
- Reports or exports the status and configuration of one or more nodes

Most of the subcommands for the `clnode` command operate in cluster mode. You can run most of these subcommands from any node in the cluster. However, the `add` and `remove` subcommands are exceptions. You must run these subcommands in noncluster mode.

When you run the `add` and `remove` subcommands, you must run them on the node that you are adding or removing. The `clnode add` command also initializes the node itself for joining the cluster. The `clnode remove` command also performs cleanup operations on the removed node.

You can omit *subcommand* only if *options* is the `-?` option or the `-V` option.

Each option has a long and a short form. Both forms of each option are given with the description of the option in OPTIONS.

The `clnode` command does not have a short form.

You can use some forms of this command in a non-global zone, referred to simply as a zone. For more information about valid uses of this command in zones, see the descriptions of the individual subcommands. For ease of administration, use this command in the global zone.

Subcommands The following subcommands are supported:

`add`
Configures and adds a node to the cluster.

You can use this subcommand only in the global zone.

You must run this subcommand in noncluster mode.

To configure and add the node, you must use the `-n` *sponsornode* option. This option specifies an existing active node as the sponsor node. The sponsor node is always required when you configure nodes in the cluster.

If you do not specify `-c` *clustername*, this subcommand uses the name of the first node that you add as the new cluster name.

The operand *node* is optional. However, if you specify an operand, it must be the host name of the node on which you run the subcommand.

Note – You must run the `claccess` command to allow the node to be added to the cluster. See the claccess(1CL) man page.

This subcommand does *not* install cluster software packages. To add a node and install cluster software packages, use the `scinstall -a` command. See the scinstall(1M) man page.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

`clear`
Cleans up or clears any remaining information about cluster nodes after you run the `remove` subcommand.

You can use this subcommand only in the global zone.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`evacuate`
Attempts to switch over all resource groups and device groups from the specified node to a new set of primary nodes.

You can use this subcommand only in the global zone.

The system attempts to select new primary nodes based on configured preferences for each group. All evacuated resource groups are not necessarily remastered by the same primary node. If one or more resource groups or device groups cannot be evacuated from the specified node, this subcommand fails. If this subcommand fails, it issues an error message and exits with a nonzero exit code. If this subcommand cannot change primary ownership of a group to another node, the original node retains primary ownership of that group.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`export`
Exports the node configuration information to a file or to the standard output (`stdout`).

You can use this subcommand only in the global zone.

If you specify the `-o` option and the name of a file, the configuration information is written to that file.

If you do not provide the `-o` option and a file name, the output is written to the standard output.

This subcommand does not modify cluster configuration data.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`list`
Displays the names of nodes that are configured in the cluster.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

If you do not specify the `node` operand, or if you specify the plus sign operand (+), this subcommand displays all node members.

You must run this subcommand in cluster mode.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand option. See the `rbac`(5) man page.

`remove`
Removes a node from the cluster.

You can use this subcommand only in the global zone.

You must run this subcommand in noncluster mode.

To remove a node from a cluster, observe the following guidelines. If you do not observe these guidelines, your removing a node might compromise quorum in the cluster.

- Unconfigure the node to be removed from any quorum devices, unless you also specify the `-F` option.
- Ensure that the node to be removed is not an active cluster member.
- Do not remove a node from a three-node cluster unless at least one shared quorum device is configured.

The subcommand attempts to remove a subset of references to the node from the cluster configuration database. If you specify the `-F` option, this subcommand attempts to remove all references to the node from the cluster configuration database.

Note – You must run the `claccess` command to allow the node to be removed from the cluster. See the claccess(1CL) man page.

This subcommand does *not* remove cluster software packages. To remove both a node and cluster software packages, use the `scinstall -r` command. See the scinstall(1M) man page.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`set`
Modifies the properties that are associated with the node that you specify.

You can use this subcommand only in the global zone.

See the `-p` option in OPTIONS.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`show`
Displays the configuration of, or information about the properties on, the specified node or nodes.

You can use this subcommand only in the global zone.

If you do not specify operands or if you specify the plus sign (+), this subcommand displays information for all cluster nodes.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`show-rev`
Displays the names of and release information about the Sun Cluster packages that are installed on a node.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

You can run this subcommand in noncluster mode and cluster mode. If you run it in noncluster mode, you can only specify the name of and get information about the node on which you run it. If you run it in cluster mode, you can specify and get information about any node in the cluster.

When you use this subcommand with `-v`, this subcommand displays the names of packages, their versions, and patches that have been applied to those packages.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`status`
Displays the status of the node or nodes that you specify or Internet Protocol (IP) Network Multipathing groups.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

If you do not specify operands or if you specify the plus sign (+), this subcommand displays the status of all cluster nodes. The status of a node can be `Online` or `Offline`.

If you specify the `-m` option with this subcommand, it displays only Solaris IP multipathing (IPMP) groups.

If you specify the verbose option `-v` with this subcommand, it displays both the status of cluster nodes and Solaris IP Network Multipathing groups.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

Options **Note –** Both the short and long form of each option is shown in this section.

The following options are supported:

`-?`
`--help`
Displays help information.

You can specify this option with or without a *subcommand*.

If you do not specify a *subcommand*, the list of all available subcommands is displayed.

If you specify a *subcommand*, the usage for that subcommand is displayed.

If you specify this option and other options, the other options are ignored.

`-c` *clustername*
`--clustername=`*clustername*
`--clustername` *clustername*
Specifies the name of the cluster to which you want to add a node.

Use this option only with the `add` subcommand.

If you specify this option, the *clustername* that you specify must match the name of an existing cluster. Otherwise, an error occurs.

`-e` *endpoint*`-,`*endpoint*
`--endpoint=`*endpoint*`-,`*endpoint*
`--endpoint` *endpoint*`-,`*endpoint*
Specifies transport connections.

Use this option only with the `add` subcommand. You specify this option to establish the cluster transport topology. You establish the topology by configuring the cables that connect the adapters and the switches. You can specify an adapter or a switch as the endpoint. To indicate a cable, you specify a comma separated pair of endpoints. The cable establishes a connection from a cluster transport adapter on the current node to one of the following:

- A port on a cluster transport switch, also called a transport junction.
- An adapter on another node that is already included in the cluster.

If you do not specify the `-e` option, the `add` subcommand attempts to configure a default cable. However, if you configure more than one transport adapter or switch within one instance of the `clnode` command, `clnode` cannot construct a default. The default is to configure a cable from the singly configured transport adapter to the singly configured, or default, transport switch.

You must always specify two endpoints that are separated by a comma every time you specify the `-e` option. Each pair of endpoints defines a cable. Each individual endpoint is specified in one of the following ways:

- Adapter endpoint:

  node:*adapter*
- Switch endpoint:

  switch[*@port*]

To specify a tagged-VLAN adapter, use the tagged-VLAN adapter name that is derived from the physical device name and the VLAN instance number. The VLAN instance number is the VLAN ID multiplied by 1000 plus the original physical-unit number. For example, a VLAN ID of 11 on the physical device `ce2` translates to the tagged-VLAN adapter name `ce11002`.

If you do not specify a port component for a switch endpoint, a default port is assigned for all but SCI switches.

`-F`
`--force`
Forcefully removes or clears the specified node without verifying that global mounts remain on that node.

Use this option only with the `clear` or the `remove` subcommand.

`-G` *globaldevfs*
`--globaldevfs=`*globaldevfs*
`--globaldevfs` *globaldevfs*
Specifies either an existing file system or a raw special disk device to use for the global devices file system.

Use this option only with the `add` or `remove` subcommand.

Each cluster node must have a local file system that is mounted globally on `/global/.devices/node@`*nodeID* before the node can successfully participate as a cluster member. However, the node ID is unknown until the `clnode` command runs. The file system that you specify is remounted at `/globaldevices`. The `clnode` command attempts to add the entry to the `vfstab` file when the command cannot find a node ID mount. See the `vfstab`(4) man page.

As a guideline, the file system that you specify must be at least 512 Mbytes in size. If this partition or file system is not available or is not large enough, you might need to reinstall the Solaris Operating System.

Use this option with the `remove` subcommand to specify the new mount point name to use to restore a former `/global/.devices` mount point.

When used with the `remove` subcommand, this option specifies the new mount point name to use to restore the former `/global/.devices` mount point. If you do not specify the `-G` option, the mount point is renamed `/globaldevices` by default.

`-i` {`-` | *clconfigfile*}
`--input={- |` *clconfigfile*`-}`
`--input {- |` *clconfigfile*`-}`
Reads node configuration information from a file or from the standard input (`stdin`). The format of the configuration information is described in the clconfiguration(5CL) man page.

If you specify a file name with this option, this option reads the node configuration information in the file. If you specify `-` with this option, the configuration information is read from the standard input (`stdin`).

`-m`
Specifies IP multipathing (IPMP) groups. Use with the `status` subcommand to display only the status of IPMP groups.

`-n` *sponsornode*
`--sponsornode=`*sponsornode*
`--sponsornode` *sponsornode*
Specifies the name of the sponsor node.

You can specify a name or a node identifier for *sponsornode*. When you add a node to the cluster by using the `add` subcommand, the sponsor node is the first active node that you add to the cluster. From that point, that node remains the *sponsornode* for that cluster. When you remove a node by using the `remove` subcommand, you can specify any active node other than the node to be removed as the sponsor node.

By default, whenever you specify *sponsornode* with a subcommand, the cluster to which *sponsornode* belongs is the cluster that is affected by that subcommand.

`-o` {`-` | *clconfigfile*}
`--output={-` | *clconfigfile*`-}`
`--output {-` | *clconfigfile*`-}`
Writes node configuration information to a file or to the standard output (`stdout`). The format of the configuration information is described in the clconfiguration(5CL) man page.

If you specify a file name with this option, this option creates a new file. Configuration information is then placed in that file. If you specify `-` with this option, the configuration information is sent to the standard output (`stdout`). All other standard output for the command is suppressed.

You can use this option only with the `export` subcommand.

`-p` *name*
`--property=`*name*
`--property` *name*
Specifies the node properties about which you want to display information with the `show` subcommand.

For information about the properties that you can add or modify with the `set` subcommand, see the description of the `-p` *name*=*value* option.

You can specify the following properties with this option:

`adapterlist`
This property specifies one or more transport adapters, each separated by a comma (,), for a node.

`defaultpsetmin`
This property specifies the minimum number of CPUs that are available in the default processor set resource. You can set this property to any value between 1 and the number of CPUs on the machine (or machines) on which this property is set.

`globalzoneshares`
This property specifies the number of shares that are assigned to the global zone. You can set this property to any value between 1 and 65535, inclusive.

`monitoring`
Values to which you can set this property are `enabled` and `disabled`.

`privatehostname`
The private host name is used for IP access of a given node over the private cluster interconnect. By default, when you add a node to a cluster, this option uses the private host name `clusternode`*nodeid*`-priv`.

`reboot_on_path_failure`
Values to which you can set this property are `enabled` and `disabled`.

`zprivatehostname`
The zones private host name is used for IP access of a given zone on a node over the private cluster interconnect.

`-p` *name=value*
`--property=`*name-=value*
`--property` *name-=value*
Specifies the node properties that you want to add or modify with the `set` subcommand.

Multiple instances of `-p` *name=value* are allowed.

For information about the properties about which you can display information with the `show` subcommand, see the description of the `-p` *name* option.

You can modify the following properties with this option:

`adapterlist`
Specifies one or more transport adapters, each separated by a comma (,), for a node.

`defaultpsetmin`
Sets the minimum number of CPUs that are available in the default processor set resource.

The default value is 1 and the minimum value is 1. The maximum value is the number of CPUs on the machine (or machines) on which you are setting this property.

`globalzoneshares`
Sets the number of shares that are assigned to the global zone.

You can specify a value between 1 and 65535, inclusive. To understand this upper limit, see the `prctl`(1) man page for information about the `zone.cpu-shares` attribute. The default value for `globalzoneshares` is 1.

`monitoring`
Enables or disables disk path monitoring on the specified node. Values to which you can set this property are `enabled` and `disabled`.

`privatehostname`
Is used for IP access of a given node over the private cluster transport. By default, when you add a node to a cluster, this option uses the private host name `clusternode`*nodeid*`-priv`.

Before you modify a private host name, you must disable, on all nodes, all resources or applications that use that private host name. See the example titled "Changing the Private Hostname" in "How to Change the Node Private Host Name" in *Sun Cluster System Administration Guide for Solaris OS*.

Do *not* store private host names in the `hosts` database or in any naming services database. See the `hosts`(4) man page. A special `nsswitch` command performs all host name lookups for private host names. See the `nsswitch.conf`(4) man page.

If you do not specify a *value*, this option uses the default private host name `clusternode`*nodeid*`-priv`.

`reboot_on_path_failure`
Enables the automatic rebooting of a node when all monitored disk paths fail, provided that the following conditions are met:

- All monitored disk paths on the node fail.
- At least one of the disks is accessible from a different node in the cluster.

You can use only the `set` subcommand to modify this property. You can set this property to `enabled` or to `disabled`.

Rebooting the node restarts all resource groups and device groups that are mastered on that node on another node.

If all monitored disk paths on a node remain inaccessible after the node automatically reboots, the node does not automatically reboot again. However, if any monitored disk paths become available after the node reboots but then all monitored disk paths again fail, the node automatically reboots again.

If you set this property to `disabled` and all monitored disk paths on the node fail, the node does *not* reboot.

`zprivatehostname`
Does the following:

- Assigns an IP address to a local zone and plumbs this IP address over the private cluster interconnect.
- Changes an IP address for a local zone and plumbs this IP address over the private cluster interconnect.
- Frees an IP address that is assigned to a local zone, unplumbs it, and makes it available for use elsewhere.

You specify a *value* as follows:

`zprivatehostname=[`*hostalias*`]` *node:zone*

hostalias
Provides the host alias to be used for accessing a zone on a node over the private cluster interconnect.

If a host alias does not exist for the *zone* on the *node*, specifying this value creates a new host alias. If a host alias already exists, specifying this value changes the existing host alias to the new host alias that you specify. If you do not specify a *hostalias*, the host alias that is assigned to *node:zone* is freed for use elsewhere.

node:zone
Provides the name or ID of a zone on a node to be assigned the private host name or host alias.

Before you modify a private host name or alias, you must disable, on all nodes, all resources or applications that use that private host name or alias. See the example titled "Changing the Private Hostname" in "How to Change the Node Private Host Name" in *Sun Cluster System Administration Guide for Solaris OS*.

If you do not specify a *value*, this option uses the default private host name `clusternode`*nodeid*`-priv`.

`-T` *seconds*
`--time=`*seconds*
`--time` *seconds*
Specifies the number of seconds to keep resource groups from switching back onto a node after you have evacuated resource groups from the node.

You can use this option only with the `evacuate` subcommand. You must specify an integer value between 0 and 65535 for *seconds*. If you do not specify a value, 60 seconds is used by default.

Resource groups cannot fail over or automatically switch over onto the node while that node is being evacuated. This option also specifies that after a node is evacuated, resource groups cannot fail over or automatically switch over for *seconds* seconds. You can, however, initiate a switchover onto the evacuated node by using the `clresourcegroup` command before *continue_evac* seconds have passed. Only automatic switchovers are prevented. See the clresourcegroup(1CL) man page.

`-v`
`--verbose`
Displays verbose information on the standard output (`stdout`).

`-V`
`--version`
Displays the version of the command.

If you specify this option with other options, with subcommands, or with operands, they are all ignored. Only the version of the command is displayed. No other processing occurs.

Operands The following operands are supported:

node The name of the node that you want to manage.

When you use the `add` subcommand, you specify the host name for *node*. When you use another subcommand, you specify the node name or node identifier for *node*.

`+` All nodes in the cluster.

Exit Status The complete set of exit status codes for all commands in this command set are listed on the Intro(1CL) man page.

If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0 CL_NOERR`
No error

The command that you issued completed successfully.

`1 CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3 CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6 CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

15 CL_EPROP
Invalid property

The property or value that you specified with the -p, -y, or -x option does not exist or is not allowed.

35 CL_EIO
I/O error

A physical input/output error has occurred.

36 CL_ENOENT
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the -o option does not exist.
- The configuration file that you attempted to access with the -i option contains errors.

37 CL_EOP
Operation not allowed

You tried to perform an operation on an unsupported configuration, or you performed an unsupported operation.

Examples

EXAMPLE 1 Adding a Node to a Cluster

The following command configures and adds the node on which you run the command into an existing cluster. By default, this example uses /globaldevices as the global devices mount point. By default, this example also uses clusternode1-priv as the private host name.

This command names the cluster cluster-1 and specifies that the sponsor node is phys-schost-1. This command also specifies that adapter qfe1 is attached to transport switch switch1. Finally, this command specifies that adapter qfe2 is attached to transport switch switch2.

```
# clnode add -c cluster-1 -n phys-schost-1 \
-e phys-schost-2:qfe1,switch1 -e phys-schost-2:qfe2,switch2
```

EXAMPLE 2 Removing a Node From a Cluster

The following command removes a node from a cluster. This command removes the node on which you run this command. The node is in noncluster mode.

```
# clnode remove
```

EXAMPLE 3 Changing the Private Host Name That Is Associated With a Node

The following command changes the private host name for node `phys-schost-1` to the default setting.

```
# clnode set -p privatehost=phys-schost-1
```

EXAMPLE 4 Changing Private Host Name Settings for All Nodes

The following command changes the private host name settings for all nodes to default values. In this case, you *must* insert a space between the equal sign (=) and the plus sign (+) to indicate that the + is the plus sign operand.

```
# clnode set -p privatehost= +
```

EXAMPLE 5 Removing the Private Host Name for a Zone That Is Associated With a Node

The following command disables or removes the private host name for zone `dev-zone`, which is associated with node `phys-schost-1`.

```
# clnode set -p zprivatehostname= phys-schost-1:dev-zone
```

EXAMPLE 6 Displaying the Status of All Nodes in a Cluster

The following command displays the status of all nodes in a cluster.

```
# clnode status
=== Cluster Nodes ===

--- Node Status ---

Node Name                                       Status
---------                                       ------
phys-schost-1                                   Online
phys-schost-2                                   Online
```

EXAMPLE 7 Displaying the Verbose Status of All Nodes in a Cluster

The following command displays the verbose status of all nodes in a cluster.

```
# clnode status -v
=== Cluster Nodes ===

--- Node Status ---

Node Name                                       Status
---------                                       ------
```

EXAMPLE 7 Displaying the Verbose Status of All Nodes in a Cluster *(Continued)*

```
phys-schost-1                                   Online
phys-schost-2                                   Online


--- Node IPMP Group Status ---

Group Name      Node Name           Adapter     Status
----------      ---------           -------     ------
sc_ipmp0        phys-schost-1       hme0        Online
sc_ipmp0        phys-schost-2       hme0        Online
```

EXAMPLE 8 Displaying Configuration Information About All Nodes in a Cluster

The following command displays configuration information about all nodes in a cluster.

```
# clnode show
=== Cluster Nodes ===

Node Name:                                      phys-schost-1
  Node ID:                                         1
  Enabled:                                         yes
  privatehostname:                                 clusternode1-priv
  reboot_on_path_failure:                          disabled
  globalzoneshares:                                1
  defaultpsetmin:                                  1
  quorum_vote:                                     1
  quorum_defaultvote:                              1
  quorum_resv_key:                                 0x4487349A00000001
  Transport Adapter List:                          ce2, bge2

Node Name:                                      phys-schost-2
  Node ID:                                         2
  Enabled:                                         yes
  privatehostname:                                 clusternode2-priv
  reboot_on_path_failure:                          disabled
  globalzoneshares:                                1
  defaultpsetmin:                                  1
  quorum_vote:                                     1
  quorum_defaultvote:                              1
  quorum_resv_key:                                 0x4487349A00000002
  Transport Adapter List:                          ce2, bge2
```

EXAMPLE 9 Displaying Configuration Information About a Particular Node in a Cluster

The following command displays configuration information about `phys-schost-1` in a cluster.

EXAMPLE 9 Displaying Configuration Information About a Particular Node in a Cluster *(Continued)*

```
# clnode show phys-schost-1
=== Cluster Nodes ===

Node Name:                                      phys-schost-1
  Node ID:                                        1
  Enabled:                                        yes
  privatehostname:                                clusternode1-priv
  reboot_on_path_failure:                         disabled
  globalzoneshares:                               1
  defaultpsetmin:                                 1
  quorum_vote:                                    1
  quorum_defaultvote:                             1
  quorum_resv_key:                                0x4487349A00000001
  Transport Adapter List:                         ce2, bge2
```

EXAMPLE 10 Displaying Configuration Information About All Nodes in a Cluster With Zones

The following command displays configuration information about all nodes in a cluster with zones.

```
# clnode show

== Cluster Nodes ===

Node Name:                                      phys-schost-1
  Node ID:                                        1
  Enabled:                                        yes
  ...
  Zone List:                                      phys-schost-1:lzphys-schost-1a
                                                  phys-schost-1:zone1
  Transport Adapter List:                         hme1, hme3

  --- Zones for phys-schost-1 ---

    Zone Name:                                  phys-schost-1:lzphys-schost-1a
      zprivatehostname                            priv_zone1a

    Zone Name:                                  phys-schost-1:zone1
      zprivatehostname                            priv_zone_1
```

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also prctl(1), claccess(1CL), clresourcegroup(1CL), cluster(1CL), Intro(1CL), newfs(1M), su(1M), hosts(4), scinstall(1M), nsswitch.conf(4), vfstab(4), attributes(5), rbac(5), clconfiguration(5CL)

The example that describes how to change the private hostname in "Overview of Administering the Cluster" in *Sun Cluster System Administration Guide for Solaris OS*

Notes The superuser can run all forms of this command.

All users can run this command with the -? (help) or -V (version) option.

To run the clnode command with subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
add	solaris.cluster.modify
clear	solaris.cluster.modify
evacuate	solaris.cluster.admin
export	solaris.cluster.read
list	solaris.cluster.read
remove	solaris.cluster.modify
set	solaris.cluster.modify
show	solaris.cluster.read
show-rev	solaris.cluster.read
status	solaris.cluster.read

Name clquorum, clq – manage Sun Cluster quorum devices and properties

Synopsis

```
/usr/cluster/bin/clquorum  -V

/usr/cluster/bin/clquorum subcommand
 -?

/usr/cluster/bin/clquorum subcommand
 [options] -v devicename[ ...]

/usr/cluster/bin/clquorum add  [-a] [-t type] [-p  name=value[,...]]
 devicename[ ...]

/usr/cluster/bin/clquorum add  -i {- |  clconfigfile}
 [-t type] [-p  name=value[,...]] {+ |  devicename[ ...]}

/usr/cluster/bin/clquorum disable  [-t  type[,...]]
 {+ |  devicename...}

/usr/cluster/bin/clquorum enable  [-t  type[,...]]
 {+ |  devicename[ ...]}

/usr/cluster/bin/clquorum export  [-o  {- |  clconfigfile}]
 [-t  type[,...]] {+ |  devicename[ ...]}

/usr/cluster/bin/clquorum list  [-t  type[,...]]
 [-n  node[,...]] [+ |  devicename[ ...]]

/usr/cluster/bin/clquorum remove  [-t  type[,...]]
 {+ |  devicename[ ...]}

/usr/cluster/bin/clquorum reset

/usr/cluster/bin/clquorum show  [-t  type[,...]]
 [-n  node[,...]] [+ |  devicename[ ...]]

/usr/cluster/bin/clquorum status  [-t  type[,...]]
 [-n  node[,...]] [+ |  devicename[ ...]]
```

Description The `clquorum` command manages cluster quorum devices and cluster quorum properties. The `clq` command is the short form of the `clquorum` command. The `clquorum` command and the `clq` command are identical. You can use either form of the command.

You can use some forms of this command in a non-global zone, referred to simply as a zone. For more information about valid uses of this command in zones, see the descriptions of the individual subcommands. For ease of administration, use this command in the global zone.

The general form of this command is as follows:

`clquorum` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the `-?` option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are provided with the description of the option in the "OPTIONS" section of this man page.

Quorum devices are necessary to protect the cluster from split-brain and amnesia situations. (For information about split-brain and amnesia situations, see the section on quorum and quorum devices in the *Sun Cluster Concepts Guide for Solaris OS*.) Each quorum device must be connected, either by a SCSI cable or through an IP network, to at least two nodes.

A quorum device can be a shared SCSI storage device, a shared NAS storage device, or a quorum server. If the quorum device stores user data, you do not affect this data if you add or remove such a device as a quorum device. However, if you are using replicated storage devices, the quorum device must be on an unreplicated volume.

Both nodes and quorum devices participate in cluster quorum formation, unless the nodes and quorum devices are in the maintenance state. If a node or a quorum device is in the maintenance state, its vote count is always zero and it does not participate in quorum formation.

You can use the `clquorum` command to perform the following tasks:

- Add a quorum device to the Sun Cluster configuration
- Remove a quorum device from the Sun Cluster configuration
- Manage quorum properties

Subcommands The following subcommands are supported:

`add`
Adds the specified shared device as a quorum device.

You can use this subcommand only in the global zone.

Each quorum device must be connected to at least two nodes in the cluster. The quorum device is added with connection paths in the cluster configuration to every node to which the device is connected. Later, if the connection between the quorum device and the cluster nodes changes, you must update the paths. Update the paths by removing the quorum device and then adding it back to the configuration. This situation could arise if you add more nodes that are connected to the quorum device or if you disconnect the quorum device from one or more nodes. For more information about quorum administration, see the Sun Cluster administration guide.

Quorum devices have several types. See the `-t` option in the "OPTIONS" section for a complete description. The `scsi` type is the default type.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization. See `rbac`(5).

See also the description of the `remove` subcommand.

`disable`
Puts a quorum device or node in the quorum maintenance state.

You can use this subcommand only in the global zone.

In the maintenance state, a shared device or node has a vote count of zero. This shared device or node no longer participates in quorum formation. In addition, for a node that is in the maintenance state, any quorum devices that are connected to the node have their vote counts decremented by one.

This feature is useful when you need to shut down a node or a device for an extended period of time for maintenance. After a node boots back into the cluster, the node removes itself from maintenance mode unless the `installmode` is set.

You must shut down a node before you can put the node in the maintenance state.

Users other than superuser require `solaris.cluster.modify` RBAC authorization. See `rbac`(5).

See also the description of the `enable` subcommand.

`enable`
Removes a quorum device or a node from the quorum maintenance state.

You can use this subcommand only in the global zone.

The `enable` subcommand removes a quorum device or node from maintenance mode. The subcommand resets the configured quorum vote count of a quorum device or node to the default. The shared device or node can then participate in quorum formation.

After resetting a quorum device, the vote count for the quorum device is changed to *N*-1. In this calculation, *N* is the number of nodes with nonzero vote counts that are connected to the device. After resetting a node, the vote count is reset to its default. Then the quorum devices that are connected to the node have their vote counts incremented by one.

Unless the install mode setting `installmode` is enabled, the quorum configuration for each node is automatically enabled at boot time.

Users other than superuser require `solaris.cluster.modify` RBAC authorization. See `rbac`(5).

See also the description of the `disable` subcommand.

`export`
Exports the configuration information for the cluster quorum.

You can use this subcommand only in the global zone.

If you specify a file by using the `-o` option, the configuration information is written to that file. If you do not specify a file, the information is written to standard output (`stdout`).

The `export` subcommand does not modify any cluster configuration data.

Users other than superuser require `solaris.cluster.read` RBAC authorization. See `rbac`(5).

`list`
Displays the names of quorum devices that are configured in the cluster.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

If you do not specify options, the `list` subcommand displays all the quorum devices that are configured in the cluster. If you specify the `-t` option, the subcommand displays only quorum devices of the specified type. If you specify the `-n` option, the subcommand displays the names of all quorum devices that are connected to any of the specified nodes.

Users other than superuser require `solaris.cluster.read` RBAC authorization. See `rbac`(5).

`remove`
Removes the specified quorum device or devices from the Sun Cluster quorum configuration.

You can use this subcommand only in the global zone.

The `remove` subcommand does not disconnect and remove the physical device. The subcommand also does not affect the user data on the device, if any data exists. The last quorum device in a two-node cluster cannot be removed, unless the `installmode` is enabled.

You can remove only a quorum device. You cannot use this subcommand to remove cluster nodes.

Users other than superuser require `solaris.cluster.modify` RBAC authorization. See `rbac`(5).

See also the description of the `add` subcommand.

`reset`
Resets the entire quorum configuration to the default vote count settings.

You can use this subcommand only in the global zone.

If `installmode` is enabled, the mode is cleared by resetting. `installmode` cannot be reset on a two-node cluster unless at least one quorum device has been successfully configured.

Users other than superuser require `solaris.cluster.modify` RBAC authorization. See `rbac`(5).

See also the `-p` option in cluster(1CL) for the description of the `installmode` property.

`show`
Displays the properties of quorum devices.

You can use this subcommand only in the global zone.

If you do not specify options, the `show` subcommand displays the properties of all the quorum devices in the cluster.

If you specify the type by using the `-t` option, the subcommand displays properties of devices of that type only. See `-t` in "OPTIONS".

If you specify nodes by using the `-n` option, this subcommand displays the properties of the quorum devices that are connected to any of the specified nodes.

Users other than superuser require `solaris.cluster.read` RBAC authorization. See `rbac`(5).

`status`
Displays the status and vote counts of quorum devices.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

If you do not specify options, the `status` subcommand displays information about all the quorum devices in the cluster.

If you specify the type by using the `-t` option, the subcommand displays information about devices of that type only. See `-t` in "OPTIONS".

If you specify nodes by using the `-n` option, this subcommand displays the properties of the quorum devices that are connected to any of the specified nodes.

Users other than superuser require `solaris.cluster.read` RBAC authorization. See `rbac`(5).

Options The following options are supported:

`-?`
`--help`
Displays help information. When this option is used, no other processing is performed.

You can specify this option without a subcommand or with a subcommand.

If you specify this option without a subcommand, the list of subcommands of this command is displayed.

If you specify this option with a subcommand, the usage options for the subcommand are displayed.

`-a`
`--autoconfig`
For a two-node cluster that uses shared SCSI devices, automatically chooses and configures one quorum device if no quorum devices are configured.

All SCSI devices in the cluster must be qualified to be a quorum device. The `autoconfig` subcommand does not check whether an available device is qualified to be a quorum device. The `autoconfig` subcommand checks only for SCSI storage devices.

Users other than superuser require `solaris.cluster.modify` RBAC authorization. See `rbac`(5).

`-i` *clconfigfile*
`--input=`*clconfigfile*
`--input` *clconfigfile*
Specifies configuration information that is to be used for managing the quorum devices. This information must conform to the format that is defined in the clconfiguration(5CL) man page.

When `-i` is used with a subcommand along with other command-line options, the arguments of the command-line options overwrite the settings in the configuration file.

`-n` *node*
`--node=`*node_name*
`--node` *node_name*
Specifies the node name to which the quorum devices are connected. This option is used in the `list`, `status`, and `show` subcommands to limit the information that is displayed to those quorum devices that are connected to the specified nodes.

You can specify either a node name or a node ID for the *node_name*.

`-o` {`-` | *clconfigfile*}
`--output={- |` *clconfigfile*`-}`
`--output {- |` *clconfigfile*`-}`
Writes quorum-device-configuration information to a file or to the standard output (`stdout`). The format of this configuration information conforms to the format that is described in the clconfiguration(5CL) man page. To specify the standard output, specify `-` instead of a file name.

`-p` *name*=*value*[,...]
`--property` *name*`-=`*value*`-[,...]`
`--property` *name*`-` *value*`-[,...]`
Specifies properties of a quorum device that are specific to a device type. You use this option with the `add` subcommand. See the description of the `-t` option for a list and a description of these properties.

`-t` *type*
`--type=`*device_type*
`--type` *device_type*
Specifies the quorum device type. When this option is specified, the operands must be of the specified type.

For the `add`, `export`, and `remove` subcommands, the current supported quorum device types are as follows:

- Shared local disks, specified by `scsi`
- Network Attached Storage device from Network Appliance, Inc., specified by `netapp_nas`
- A quorum server process that runs on the Sun Cluster Quorum Server machine, specified by `quorum_server`

The default type is `scsi`.

The `add` subcommand does not accept `-tnode` as a quorum type.

For the `enable`, `disable`, `list`, `show`, and `status` subcommands, the type can be `node`, `scsi`, `netapp_nas`, or `quorum_server`. These different types of quorum devices have the following properties:

`node`
: No specific properties are set for nodes to participate in quorum formation.

  This type is used only with the `enable`, `disable`, `list`, `status`, and `show` subcommands. It cannot be used to add a quorum device of type `node`.

`scsi`
: No specific properties are set for `scsi` quorum devices. The `autoconfig` subcommand accepts only this quorum device type.

`netapp_nas`
: The `netapp_nas` type of quorum device has the following properties:

  `filer=`*filer-name*: Specifies the name of the device on the network. The cluster uses this name to access the NAS device.

  `lun_id=`*lun-id*: Specifies the logical unit number (LUN) on the NAS device that will be a NAS quorum device. The `lun_id` property defaults to 0. If you have configured LUN 0 on your device for the quorum device, you do not need to specify this property.

  These properties are required when using the `add` subcommand to add a NAS device as a quorum device.

  Before you can add a quorum device, the NAS device must be set up and operational. The NAS device must be already booted and running, and the LUN to be used as a NAS quorum must be already created.

To provide support for NAS devices as quorum devices, the cluster administrator must install the quorum device support module that Network Appliance, Inc. provides. The `add` subcommand fails if the module is not installed on the cluster nodes. See *Sun Cluster With Network-Attached Storage Devices Manual for Solaris OS* for instructions about obtaining the support module. Additionally, the iSCSI license must be valid for the Network Appliance, Inc. NAS device.

After the cluster administrator performs the required procedures, you can use the `clquorum add` subcommand to add the NAS device as a quorum device.

`quorum_server`

The `quorum_server` type of quorum device has the following properties:

`qshost=`*quorum-server-host*: Specifies the name of the machine where the quorum server runs. This host can be the IP address of the machine or the hostname on the network. If you specify the hostname, the IP address of the machine must be specified in the `/etc/hosts` file, the `/etc/inet/ipnodes` file, or both.

`port=`*port*: Specifies the port number used by the quorum server to communicate with the cluster nodes.

Before you can add a quorum server, the quorum server software must be installed on the host machine and the quorum server must be started and running. Refer to the *Sun Cluster Quorum Server User Guide* for details.

`-V`
`--version`

Displays the version of the command.

Do not specify this option with other subcommands, options, or operands. The subcommands, options, or operands are ignored. The `-V` option displays only the version of the command. No other operations are performed.

`-v`
`--verbose`

Displays verbose information to standard output (`stdout`).

Operands The following operands are supported:

devicename
For the `add`, `export`, and `remove` subcommands only, the operand is the name of a SCSI, quorum server, or NAS quorum device. For the `add` subcommand, if you do not specify a `clconfigurationfile` by using `-i`, you must specify at least one quorum device as the operand.

For the `disable`, `enable`, `list`, `status`, and `show` subcommands only, the operand can be the name of a node or of a SCSI, quorum server, or NAS quorum device.

In every case, the operand type must match the value of the `-t` option, if you specify that option.

Use the following values as the *devicename* operand:

- For nodes, the operand must be the node name or the node ID.
- For SCSI quorum devices, the operand must be the device identifier or the full DID path name, for example, `d1` or `/dev/did/rdsk/d1`.
- For NAS quorum devices, the operand must be the device name as defined when you added the device to the cluster configuration.
- For quorum server quorum devices, the operand must specify an identifier for the quorum server or servers. This can be the quorum server instance name, and must be unique across all quorum devices.

+
For the `disable`, `enable`, `list`, `status`, and `show` subcommands only, specifies all quorum devices configured for the cluster. If you use the `-t` option, the plus sign (+) operand specifies all devices of that type.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

The following exit values can be returned:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

6 CL_EACCESS
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the su(1M) and rbac(5) man pages for more information.

18 CL_EINTERNAL
Internal error was encountered

An internal error indicates a software defect or other defect.

35 CL_EIO
I/O error

A physical input/output error has occurred.

36 CL_ENOENT
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the -o option does not exist.
- The configuration file that you attempted to access with the -i option contains errors.

39 CL_EEXIST
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

41 CL_ETYPE
Invalid type

The type that you specified with the -t or -p option does not exist.

Examples

EXAMPLE 1 Adding a SCSI Quorum Device

The following clquorum command configures a SCSI quorum device that is connected to all the cluster nodes.

```
# clquorum add /dev/did/rdsk/d4s2
```

When you use the add subcommand, the scsi type is the default. To add a scsi quorum device, you do not need to specify -t scsi.

EXAMPLE 2 Adding a Network Appliance NAS Quorum Device

The following `clquorum` command adds the Network Appliance NAS quorum device `qd1` that is connected to all the cluster nodes.

```
# clquorum -t netapp_nas -p filer=nas1.sun.com -p lun_id=0 qd1
```

The name of the NAS quorum device must be unique across all cluster quorum devices.

EXAMPLE 3 Adding a Quorum Server

The following `clquorum` command configures a quorum server, `qs1`:

```
# clquorum add -t quorum_server -p qshost=10.11.114.81 -p port=9000 qs1
```

EXAMPLE 4 Removing a Quorum Device

The following `clquorum` command removes the `d4` quorum device.

```
# clquorum remove d4
```

The command that you use to remove a quorum device is the same, whether your device has a type of `scsi`, `nas`, or `quorum_server`.

EXAMPLE 5 Putting a Quorum Device into a Maintenance State

The following `clquorum` command puts a quorum device, `qs1` into a maintenance state and verifies that the device is in a maintenance state.

```
# clquorum disable qs1
# clquorum status qs1

=== Cluster Quorum ===

--- Quorum Votes by Device ---

Device Name       Present      Possible      Status
-----------       -------      --------      ------
qs1                1            1             Offline
```

EXAMPLE 6 Resetting the Quorum Votes of a Quorum Device

The following `clquorum` command resets the configured quorum vote count of a quorum device, `d4`, to the default.

```
# clquorum enable d4
```

EXAMPLE 7 Displaying the Configured Quorum Devices in the Cluster

The following `clquorum` commands display the quorum devices in concise format and verbose format.

```
# clquorum list
d4
qd1
pcow1
pcow2

# clquorum list -v
Quorums               Type
-------               ----
d4    scsi
qd1                   netapp_nas
pcow1                 node
pcow2                 node
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cluster(1CL), clconfiguration(5CL)

Notes

The superuser can run all forms of this command.

Any user can run this command with the following options:

- `-?` option
- `-V` option

To run this command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`add`	`solaris.cluster.modify`
`disable`	`solaris.cluster.modify`
`enable`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`

Subcommand	RBAC Authorization
list	solaris.cluster.read
remove	solaris.cluster.modify
reset	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read

Name clquorumserver, clqs – manage quorum servers

Synopsis

```
/usr/cluster/bin/clquorumserver -V

/usr/cluster/bin/clquorumserver subcommand
 -?

/usr/cluster/bin/clquorumserver subcommand
 [-v] [quorumserver]

/usr/cluster/bin/clquorumserver clear  -c clustername
 -I clusterID [-y] quorumserver

/usr/cluster/bin/clquorumserver show  [+ |  quorumserver...]

/usr/cluster/bin/clquorumserver start  + |  quorumserver...

/usr/cluster/bin/clquorumserver stop  + |  quorumserver...
```

Description Use the `clquorumserver` command for the following tasks:

- to clean up stale configuration information of one or more quorum servers
- to display the configuration of one or more quorum servers
- to start one or more quorum servers
- to stop one or more quorum servers

The `clqs` command is the short form of the `clquorumserver` command. You can use either form of the command.

The general form of this command is as follows:

`clquorumserver` [*subcommand*] [*options*]

You can omit *subcommand* only if *options* specifies the `-?`, `-v`, or `-V` options.

The quorum server must be configured as a quorum device for the cluster. For information about configuring the quorum server, see scqsd.conf(4) and scqsd(1M). For information about adding a `quorum_server` type of quorum device to the cluster, see clquorum(1CL).

Subcommands The following subcommands are supported:

`clear`

Removes outdated cluster information from the quorum server. The quorum server keeps information about the cluster which it serves as a quorum device. This information can become invalid in the following circumstances:

- When a cluster is decommissioned without first removing the cluster quorum device using the `clquorum remove` command
- When a `quorum_server` type quorum device is removed from a cluster while the quorum server host is down

Caution – If a quorum server is not yet removed from the cluster, using this subcommand to clean up a valid quorum server could compromise the cluster quorum.

You must specify the cluster name and cluster ID for a particular quorum server. See the `-c` and `-I` options for details.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See `rbac`(5) for more information.

`show`
Displays the configuration information about the quorum server. For every cluster that configured the quorum server as a quorum device, this subcommand shows the corresponding cluster name, cluster ID, list of reservation keys, and list of registration keys.

You can use the plus sign (+) to specify more than one quorum server.

If no operand is given, or if the plus sign (+) is specified with the operand, the command prints the configuration of all running quorum servers.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See `rbac`(5) for more information.

`start`
Starts the quorum server

`stop`
Stops the quorum server

Options The following options are supported:

`-?`
`--help`
Prints help information.

This option can be used alone or with a subcommand.

- If you use this option alone, the list of available subcommands is printed.
- If you use this option with a subcommand, the usage options for that subcommand are printed.

When this option is used, no other processing is performed.

`-c` *clustername*
`--clustername` *clustername*
Specifies the name of the cluster that uses the quorum server as a quorum device. You can get the cluster name by running some Sun Cluster commands on the cluster nodes, such as `cluster show`.

This option is required with the `clear` subcommand.

`-I` *clusterID*
`--clusterID` *clusterID*

Specifies the cluster ID. The cluster ID is an 8-digit hexadecimal number. You can get the cluster ID by running some Sun Cluster commands on the cluster nodes, such as `cluster show`.

This option is required with the `clear` subcommand.

`-V`
`--version`
Prints the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommand, operands, or other options are ignored. The `-V` option only prints the version of the command. No other operations are performed.

`-v`
`---verbose`
Prints verbose information to standard output, `stdout`.

You can specify this option with any form of this command.

`-y`
`--yes`
This option is only used with the `clear` subcommand. It pre-answers the confirmation question that is prompted by `clear` subcommand. Without this option, the `clear` subcommand asks a question to confirm whether you want to clean up the cluster information, and you need to answer `yes` or `no`. The subcommand only continues processing if you answer `yes`. When use this option, the `clear` subcommand will not ask any confirmation question, and instead directly removes the cluster information for the specified quorum server.

Operands The following operand is supported:

quorumserver
Specifies an identifier for the quorum server or servers. A quorum server can be identified by either a port number or an instance name. The port number is used by the cluster nodes to communicate with the quorum server. The instance name can be specified in the quorum server configuration file, `/etc/scqsd/scqsd.conf`. See scqsd.conf(4).

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

The following exit values are returned:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

1 CL_ENOMEM
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

3 CL_EINVAL
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the -i option was incorrect.

6 CL_EACCESS
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the su(1M) and rbac(5) man pages for more information.

18 CL_EINTERNAL
Internal error was encountered

An internal error indicates a software defect or other defect.

35 CL_EIO
I/O error

A physical input/output error has occurred.

36 CL_ENOENT
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the -o option does not exist.
- The configuration file that you attempted to access with the -i option contains errors.

Examples

EXAMPLE 1 Displaying the Configuration of One Quorum Server

The following command displays the configuration information for the quorum server that uses port 9000.

```
# clquorumserver show 9000
```

EXAMPLE 2 Displaying the Configuration of Several Quorum Servers

The following command displays the configuration information for the quorum servers listed by their instance names.

```
# clquorumserver show qs1 qs2 qs3
```

EXAMPLE 3 Displaying the Configuration of All Running Quorum Servers

The following command displays the configuration information of all running quorum servers.

```
# clquorumserver show +
```

EXAMPLE 4 Starting Quorum Servers

The following command starts all the configured quorum servers.

```
# clquorumserver start +
```

The following command starts a quorum server that is listening on port 9000.

```
# clquorumserver start 9000
```

The following command starts the quorum server instance `qs1`.

```
# clquorumserver start qs1
```

EXAMPLE 5 Stopping a Quorum Server By Port Number

The following command stops a quorum server that is listening on port 9000.

```
# clquorumserver stop 9000
```

EXAMPLE 6 Cleaning Up Outdated Cluster Information From the Quorum Server

This example removes information about the cluster named `sc-cluster` from the quorum server.

Use caution when unconfiguring a quorum server in this way. Generally, you should use `clquorum remove` to remove the quorum server device from cluster configuration and clean up the configuration information on the quorum server in a single set. You should only need to use this command if communications were lost between the cluster and the quorum server host during the `clquorum remove` operation.

```
# clquorumserver clear -c sc-cluster -I 0x4308D2CF 9000
The quorum server to be unconfigured must have been removed from
the cluster. Unconfiguring a valid quorum server could compromise
the cluster quorum. Do you want to continue? (yes or no)
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also Intro(1CL), clquorum(1CL), cluster(1CL), scqsd(1M), scqsd.conf(4).

Notes The superuser can run all forms of this command.

Any user can run this command with the following options:

- `-?` (help) option
- `-V` (version) option

To run this command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`clear`	`solaris.cluster.admin`
`show`	`solaris.cluster.read`
`start`	`solaris.cluster.admin`
`stop`	`solaris.cluster.admin`

Name clreslogicalhostname, clrslh – manage resources for Sun Cluster logical hostnames

Synopsis

```
/usr/cluster/bin/clreslogicalhostname  [subcommand]
 -?

/usr/cluster/bin/clreslogicalhostname  -V

/usr/cluster/bin/clreslogicalhostname  [subcommand
 [options]] -v  [lhresource]...

/usr/cluster/bin/clreslogicalhostname create  -g resourcegroup
 [-h lhost[,...]] [-N netif@node[,...]] [-p
 name=value] [-d] lhresource

/usr/cluster/bin/clreslogicalhostname create  -i
 {- |  clconfiguration} [-a] [-g resourcegroup[,...]]
 [-p name=value] [-d] {+ |  lhresource...}

/usr/cluster/bin/clreslogicalhostname delete  [-g resourcegroup[,...] ]
 [-F] {+ |  lhresource...}

/usr/cluster/bin/clreslogicalhostname disable  [-g resourcegroup[,...]]
 [-R] [ [-z zone]  -n node[:zone][,...]] {+ |  lhresource...}

/usr/cluster/bin/clreslogicalhostname enable  [-g resourcegroup[,...]]
 [-R] [ [-z zone]  -n node[:zone][,...]] {+ |  lhresource...}

/usr/cluster/bin/clreslogicalhostname export  [-o
 {- |  configfile}] [+ |  lhresource...]

/usr/cluster/bin/clreslogicalhostname list  [-s state[,...]]
 [-g resourcegroup[,...]] [+ |  lhresource...]

/usr/cluster/bin/clreslogicalhostname list-props
 [-l listtype] [-p name[,...] ] [+ |  lhresource...]

/usr/cluster/bin/clreslogicalhostname monitor  [-g resourcegroup[,...] ]
 {+ |  lhresource...}

/usr/cluster/bin/clreslogicalhostname reset  [-f errorflag]
 [-g resourcegroup[,...] ] {+ |  lhresource...}

/usr/cluster/bin/clreslogicalhostname set  [-i
 {- |  clconfiguration}] [-g resourcegroup[,...] ]
 [-p name[+|-]=value] {+ |  lhresource...}

/usr/cluster/bin/clreslogicalhostname show  [-g resourcegroup[,...] ]
 [-p name[,...]] [+ |  lhresource...]

/usr/cluster/bin/clreslogicalhostname status  [-s state[,...]]
 [ [-z zone]  -n node[:zone][,...]] [-g resourcegroup[,...]]
 [+ |  lhresource...]

/usr/cluster/bin/clreslogicalhostname unmonitor
 [-g resourcegroup[,...] ] {+ |  lhresource...}
```

Description The `clreslogicalhostname` command manages resources for Sun Cluster logical hostnames. The `clrslh` command is the short form of the `clreslogicalhostname` command. The `clreslogicalhostname` command and the `clrslh` command are identical. You can use either form of the command.

The `clreslogicalhostname` command includes built-in convenience options for creating logical hostname resources. The `clreslogicalhostname` command also supports the automatic creation of Solaris IP multipathing (IPMP) groups.

You can run the `clreslogicalhostname` command with the `create` subccommand or the `delete` subcommand only from the global zone.

Some subcommands of the `clreslogicalhostname` command modify the resource configuration. You can use these subcommands from the global zone or a non-global zone. If you use a subcommand that modifies resource configuration from a non-global zone, only resources that the non-global zone can master are modified. The following subcommands modify resource configuration:

- `disable`
- `enable`
- `monitor`
- `reset`
- `set`
- `unmonitor`

Some subcommands of the `clreslogicalhostname` command only obtain information about resources. You can use these subcommands from the global zone or a non-global zone. However, even when you use these subcommands from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources. The following commands only obtain information about resources:

- `export`
- `list`
- `list-props`
- `show`
- `status`

To avoid unpredictable results from this command, run all forms of the command from the global zone.

The general form of this command is as follows:

`clreslogicalhostname` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the option `-?`, `-o`, `-V`, or `-v`.

Each option of this command has a long form and a short form. Both forms of each option are given with the description of the option in the OPTIONS section of this man page.

Subcommands The following subcommands are supported:

`create`
Creates the logical hostname resources that are specified as operands to the command.

When you use `create` with the `-i` option to specify a configuration file, the subcommand accepts the plus sign (+) as an operand. When you use the + operand, all resources in the configuration file that do not exist are created.

By default, resources are created in the enabled state with monitoring enabled. However, a resource comes online and is monitored only after the resource's resource group is brought online. To create resources in the disabled state, specify the `-d` option.

You can use this subcommand only in the global zone.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand.

See also the description of the `delete` subcommand.

`delete`
Deletes the logical hostname resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are deleted.

The `-g` option filters the list of operands to limit the resources that are deleted. The `-g` option deletes only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

- By default, a resource is deleted *only* if the following conditions are met:
- The resource is disabled.
- All dependencies on the resource are eliminated.
- To ensure that all specified resources are deleted, specify the `-F` option. The effects of the `-F` option are as follows:
- All specified resources are deleted, even resources that are not disabled.
- All specified resources are removed from resource-dependency settings of other resources.

Resources are deleted in the order that is required to satisfy dependencies between the resources, regardless of the order in which resources are specified on the command line.

You can use this subcommand only in the global zone.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

See also the description of the `create` subcommand.

`disable`
Disables the logical hostname resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are disabled.

The `-g` option filters the list of operands to limit the resources that are disabled. The `-g` option disables only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

To ensure that all required resource dependencies are satisfied, specify the `-R` option. The `-R` option disables any resources that depend on the resources that are specified as operands to the command, even if the resources are not specified as operands to the command. The `-g` option and the `-t` option do not apply to resources that are to be disabled solely to satisfy resource dependencies.

Resources are disabled in the order that is required to satisfy dependencies between the resources, regardless of the order in which resources are specified on the command line.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `enable` subcommand.

`enable`
Enables the logical hostname resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are enabled.

The `-g` option filters the list of operands to limit the resources that are enabled. The `-g` option enables only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

To ensure that all required resource dependencies are satisfied, specify the `-R` option. The `-R` option enables any resources that depend on the resources that are specified as operands to the command, even if the resources are not specified as operands to the command. The `-g` option does not apply to resources that are to be enabled solely to satisfy resource dependencies.

Resources are enabled in the order that is required to satisfy dependencies between the resources, regardless of the order in which resources are specified on the command line.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `disable` subcommand.

`export`
Exports the logical hostname resource configuration in the format that is described by the clconfiguration(5CL) man page.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list`
Displays a list of the logical hostname resources that are specified as operands to the command. By default, all resources are displayed.

The `-g` option filters the list of operands to limit the resources that are displayed. The `-g` option displays only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

This subcommand accepts the plus sign (+) as an operand to specify all resources in the specified resource groups or that are instances of the specified resource types. If no operands are supplied, all resources in the specified resource groups or that are instances of the specified resource types are displayed.

If you specify the `-v` option, the resource group and resource type of each resource in the list is also displayed.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list-props`
Displays a list of the properties of the logical hostname resources that are specified as operands to the command. By default, the extension properties of all resources are displayed.

The following options filter the list of operands to limit the resources whose properties are displayed:

`-g` *resourcegrouplist*	Displays the properties only of the logical hostname resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

The `-l` option specifies the type of resource properties that are to be displayed:

`-l all`	Specifies that standard properties and extension properties are displayed.
`-l extension`	Specifies that only extension properties are displayed. By default, only extension properties are displayed.
`-l standard`	Specifies that only standard properties are displayed.

If you do not specify the `-l`option, only extension properties are displayed, unless you specify a standard property explicitly by using the `-p` option or the `-y` option.

The `-p` option limits the set of resource properties that is to be displayed. The `-p` option displays only the properties that are specified in *namelist*. You can specify standard properties and extension properties in *namelist*.

If you specify the `-v` option, the description of each property is also displayed.

This subcommand accepts the plus sign (+) as an operand to specify all resources in the specified resource groups or that are instances of the specified resource types. If no operands are supplied, properties of all resources in the specified resource groups or that are instances of the specified resource types are displayed.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`monitor`

Turns on monitoring for the logical hostname resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that monitoring is turned on for all resources.

The `-g` option filters the list of operands to limit the resources that are monitored. The `-g` option monitors only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

- If monitoring is turned on for a resource, the resource is monitored only if the following conditions are met:
- The resource is enabled.

- The resource group that contains the resource is online on at minimum one cluster node.

Note – When you turn on monitoring for a resource, you do *not* enable the resource.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `unmonitor` subcommand.

`reset`
Clears an error flag that is associated with the logical hostname resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that the error flag is cleared for all resources.

The `-g` option filters the list of operands to limit the resources that are reset. The `-g` option resets only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

By default, the `reset` subcommand clears the `STOP_FAILED` error flag. To specify explicitly the error flag that is to be cleared, use the `-f` option. The only error flag that the `-f` option accepts is the `STOP_FAILED` error flag.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

`set`
Modifies specified properties of the logical hostname resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that the specified properties of all resources are modified.

The `-g` option filters the list of operands to limit the resources that are modified. The `-g` option modifies only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`show`

Displays the configuration of the logical hostname resources that are specified as operands to the command. By default, the configuration of all resources is displayed.

The `-g` option filters the list of operands to limit the resources for which the configuration is displayed. The `-g` option displays the configuration of only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

The `-p` option limits the set of resource properties that is to be displayed. The `-p` option displays only the properties that are specified in *namelist*. You can specify standard properties and extension properties in *namelist*.

This subcommand accepts the plus sign (+) as an operand to specify all resources in the specified resource groups or that are instances of the specified resource types. If no operands are supplied, the configuration of all resources in the specified resource groups or that are instances of the specified resource types are displayed.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`status`

Displays the status of the logical hostname resources that are specified as operands to the command. By default, the status of all resources is displayed.

The following options filter the list of operands to limit the list of resources for which the status is displayed:

`-g` *resourcegrouplist*	Displays the status of only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.
`-n` *nodelist*	Displays the status of only the resources in the list of operands that are hosted on the nodes or in the zones in *nodelist*.
`-s` *statelist*	Displays the status of only the resources in the list of operands that are in the states in *statelist*.

This subcommand accepts the plus sign (+) as an operand to specify all resources in the specified resource groups or that are instances of the specified resource types. If no operands are supplied, the status of all resources in the specified resource groups or that are instances of the specified resource types are displayed.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`unmonitor`

Turns off monitoring for the logical hostname resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that monitoring is turned off for all resources.

If you turn off monitoring for a resource that is disabled, the resource is not affected. The resource and its monitor are already offline.

Note – When you turn off monitoring for a resource, you do *not* disable the resource. However, when you disable a resource, you do not need to turn off monitoring for the resource. The disabled resource and its monitor are kept offline.

The `-g` option filters the list of operands to limit the resources for which monitoring is turned off. The `-g` option turns off monitoring for the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `disable` subcommand and the `monitor` subcommand.

Options The following options are supported:

`-?`

`--help`

Displays help information. When this option is used, no other processing is performed.

You can specify this option without a subcommand or with a subcommand.

If you specify this option without a subcommand, the list of subcommands for this command is displayed.

If you specify this option with a subcommand, the usage options for the subcommand are displayed.

The effect of this option with specific subcommands is as follows:

`create`	When specified with the `-g` option, this option displays help information for all resource properties of the specified resource group.
`set`	Displays help information for properties of the resources that are specified as operands to the command.

`-a`
`--automatic`
Automatically performs the following additional operations when resources are being created from cluster configuration information:

- Registering resource types
- Creating resource groups
- Creating resources on which the resources that are specified in the list of operands depend
- The cluster configuration information must contain sufficient information to do all of the following:
- Enable the resource types to be registered
- Enable the resource groups to be created
- Enable the resources to be created

You can specify this option only with the `create` subcommand. If you specify this option, you must also specify the `-i` option and provide a configuration file.

`-d`
`--disable`
Disables a resource when the resource is created. You can specify this option only with the `create` subcommand. By default, resources are created in the enabled state.

Enabling a resource does not guarantee that the resource is brought online. A resource comes online only after the resource's resource group is brought online on at minimum one node.

`-f` *errorflag*
`--flag` *errorflag*
Specifies explicitly the error flag that is to be cleared by the `reset` subcommand. You can specify this option only with the `reset` subcommand. By default, the `reset` subcommand clears the `STOP_FAILED` error flag.

The only error flag that the `-f` option accepts is the `STOP_FAILED` error flag.

`-F`
`--force`
Forces the deletion of resources that are not disabled. You can specify this option only with the `delete` subcommand.

`-g` *resourcegroup*[,…]
`--resourcegroup` *resourcegroup*`-[,…]`
Specifies a resource group or a list of resource groups.

For subcommands except `create`, the command acts on only the resources in the list of operands that are members of the resource groups that the `-g` option specifies.

The effect of this option with specific subcommands is as follows:

`create`	Specifies that the resource is created in the specified resource group. When you use `-g` with the `create` subcommand, you can specify only one resource group.

`-h` *lhost*[,…]
`--logicalhost` *lhost*`-[,…]`
Specifies the list of logical hostnames that this resource represents. You must use the `-h` option either when more than one logical hostname is to be associated with the new logical hostname resource or when the logical hostname does not have the same name as the resource itself. All logical hostnames in the list must be on the same subnet. If you do not specify the `-h` option, the resource represents a single logical hostname whose name is the name of the resource itself.

You can use `-h` instead of setting the `HostnameList` property with `-p`. However, you cannot use `-h` and explicitly set `HostnameList` in the same command.

You can only use `-h` is with the `create` subcommand.

`-i` {`-` | *clconfiguration*}
`--input {-` | *clconfiguration*`-}`
Specifies configuration information that is to be used for creating or modifying logical hostname resources. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through the standard input. To specify the standard input, specify `-` instead of a file name.

Only the resources that are supplied as operands to the command are created or modified. Options that are specified in the command override any options that are set in the configuration information. If configuration parameters are missing in the configuration information, you must specify these parameters on the command line.

The effect of this option with specific subcommands is as follows:

`create`	When specified with the `-a` option, this option registers all required resource types and creates all required resource groups. You must supply all information that is required for the registration and configuration. All other configuration data is ignored.

`-l` *listtype*
`--listtype` *listtype*

Specifies the type of resource properties that are to be displayed by the `list-props` subcommand. You can specify this option only with the `list-props` subcommand.

You must specify one value from the following list for *listtype*:

`all`	Specifies that standard properties and extension properties are displayed.
`extension`	Specifies that only extension properties are displayed. By default, only extension properties are displayed.
`standard`	Specifies that only standard properties are displayed.

If you do not specify the `-l` option, only extension properties are displayed, unless you specify a standard property explicitly by using the `-p` option.

`-n` *node*[:*zone*][,…]
`--node` *node*`-[:`*zone*`-][,…]`
Specifies a node or a list of nodes. You can specify each node as node name or a node ID. For each node, you can optionally specify a non-global zone on the node.

The subcommands with which you can specify this option are as follows:

`disable`	Disables only the resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`enable`	Enables only the resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`status`	Reports the status only of resources in the list of operands that are hosted on the specified nodes or in the specified zones.

`-N` *netif*`@`*node*[,…]
`--netiflist` *netif*`-@`*node*`-[,…]`
Specifies a resource property. The `-N` option enables you to set the `NetIfList` property without using the `-p` option for the property. If you do not specify `-N`, the `clreslogicalhostname` command attempts to set the `NetIfList` property for you based on available IPMP groups or public adapters, and the subnet associated with the `HostnameList` property.

You can specify the `NetIfList` property in the form of *ipmpgroup*`@`*node*[,…]. However, `-N` accepts both *ipmpgroup*`@`*node*[,…] and *publicNIC*`@`*node*[,…]. If you do not use `-N`, or if you use it with *publicNIC*`@`*node*, the `clreslogicalhostname` command attempts to create the necessary IPMP groups. The system creates a set of one or more single-adapter IPMP groups with a set of default later modified to include multiple adapters using standard Solaris interfaces.

You can use `-N` instead of directly setting the `NetIfList` property with `-p`. However, you cannot use `-N` and explicitly set `NetIfList` in the same command.

You can only use `-N` with the `create` subcommand.

`-o` {`-` | *clconfiguration*}
`--output {- | clconfiguration-}`
Specifies the location where resource configuration information is to be written. This location can be a file or the standard output. To specify the standard output, specify `-` instead of a file name. If you specify the standard output, all other standard output for the command is suppressed. You can specify this option only with the `export` subcommand.

Configuration information is written only for the resources that are supplied as operands to the command. The information is written in the format that is defined in the clconfiguration(5CL) man page.

`-p` *name=value*
`-p` *name-+=array-values*
`-p` *name-=array-values*
`--property` *name-=value*
`--property` *name+=value-values*
`--property` *name--=value-values*
Sets the standard properties and extension properties of a resource. You can specify this option only with the `create` subcommand and the `set` subcommand.

For a description of standard properties, see the r_properties(5) man page.

For a description of a resource type's extension properties, see the documentation for the resource type.

The operators to use with this option are as follows:

`=` Sets the property to the specified value. The `create` subcommand and the `set` subcommand accept this operator.

`+=` Adds a value or values to a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for string array values.

`-=` Removes a value or values from a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for string array values.

If a per-node property is to be set only on a subset of cluster nodes, specify the nodes the where the property is set by appending the list of nodes in braces to the property name as follows:

name{*nodelist*}

nodelist is a comma-separated list of node names or node IDs. For more information about per-node properties, see the rt_properties(5) man page.

`-p` *name*[,…]
`--property` *name-[,…]*
Specifies a list of properties for the `list-props` subcommand and `show` subcommand.

You can use this option for standard properties and extension properties of a resource.

For a description of standard properties, see the r_properties(5) man page.

For a description of a resource type's extension properties, see the documentation for the resource type.

Without this option, the `list-props` subcommand and `show` subcommand list all or most resource properties, depending on whether the `-v` option is also specified.

`-R`
`--recursive`
Recursively enables or disables resources to ensure that all required dependencies are satisfied. You can specify this option only with the `disable` subcommand and the `enable` subcommand.

The effect of this option with these subcommands is as follows:

`disable`	Disables any resources that depend on the resources that are specified as operands to the command, even if the resources are not specified as operands to the command.
`enable`	Enables any resources on which resources that are specified as operands to the command depend, even if the resources are not specified as operands to the command.

`-s` *state*[,…]
`--state` *state*`-[,…]`
Specifies a list of states for the `list` subcommand and `status` subcommand.

This option limits the output to include only those resources that are in one of the specified states on one or more nodes in the node list.

The possible states are as follows:

- `degraded`
- `detached`
- `faulted`
- `monitor_failed`
- `not_online` - specifies any state other than `online` or `online_not_monitored`
- `offline`
- `online`
- `online_not_monitored`
- `start_failed`
- `stop_failed`
- `unknown`
- `unmonitored`
- `wait`

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The `-V` option only displays the version of the command. No other operations are performed.

`-v`
`--verbose`
Displays verbose messages to standard output.

You can specify this option with any form of the command.

Do not specify the `-v` option with the `-o -` option. The `-v` option is ignored. The `-o -` option suppresses all other standard output.

`-z` *zone*
`--zone` *zone*
Applies the same zone name to all nodes in a node list for which a zone is not explicitly specified. You can specify this option only with the `-n` option.

Operands The following operand is supported:

resource Specifies that the Sun Cluster resource names should be accepted as operands. If the subcommand accepts more than one resource, you can use the plus sign (+) to specify all logical hostname resources.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`9` `CL_ESTATE`
Object is in wrong state

You tried to modify a property, a resource group, or other object that you cannot modify at that particular time or at any time.

`10` `CL_EMETHOD`
Resource method failed

A method of a resource failed. The method failed for one of the following reasons:

- The `validate` method failed when you tried to create a resource or modify the properties of a resource.
- A method other than `validate` failed when you tried to enable, disable, or delete a resource.

`15` `CL_EPROP`
Invalid property

The property or value that you specified with the `-p`, `-y`, or `-x` option does not exist or is not allowed.

`35` `CL_EIO`
I/O error

A physical input/output error has occurred.

`36` `CL_ENOENT`
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

`39` `CL_EEXIST`
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

These exit values are compatible with the return codes that are described in the scha_calls(3HA) man page.

Examples

EXAMPLE 1 Creating a Logical Hostname Resource

This command creates a resource that is named `logicalhost1` in a resource group that is named `rg-failover`. The resource is created in the enabled state, with monitoring enabled.

```
# clreslogicalhostname create -g rg-failover logicalhost1
```

EXAMPLE 2 Creating a Logical Hostname Resource with a Different Logical Hostname

This command creates a resource named `rs-logicalhost1` in a resource group that is named `rg-failover`.

The logical hostname is not the same as the resource name, but the name and IP address of the logical host remain the same.

```
# clreslogicalhostname create -g rg-failover \
-h logicalhost1 rs-logicalhost1
```

EXAMPLE 3 Specifying the IPMP Groups for a Logical Hostname Resource

This command sets the IPMP groups for the `logicalhost1` resource.

```
# clreslogicalhostname create -g rg-failover \
-N ipmp0@black,ipmp0@white logicalhost1
```

EXAMPLE 4 Deleting a Logical Hostname Resource

This command deletes a resource that is named `logicalhost1`.

```
# clreslogicalhostname delete logicalhost1
```

EXAMPLE 5 Listing Logical Hostname Resources

This command lists all logical hostname resources.

```
# clreslogicalhostname list
logicalhost1
logicalhost2
```

EXAMPLE 6 Listing Logical Hostname Resources With Resource Groups and Resource Types

This command lists all logical hostname resources with their resource groups and resource types.

```
# clreslogicalhostname list -v
Resources    Resource Groups Resource Types
---------    --------------- --------------
```

EXAMPLE 6 Listing Logical Hostname Resources With Resource Groups and Resource Types *(Continued)*

```
logicalhost1 rg-failover-1   SUNW.LogicalHostname
logicalhost2 rg-failover-2   SUNW.LogicalHostname
```

EXAMPLE 7 Listing Extension Properties of Logical Hostname Resources

This command lists the extension properties of all logical hostname resources.

```
# clreslogicalhostname list-props -v
Properties       Descriptions
----------       ------------
NetIfList        List of IPMP groups on each node
HostnameList     List of hostnames this resource manages
CheckNameService Name service check flag
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cluster(1CL), clresource(1CL), clressharedaddress(1CL), clresourcegroup(1CL), clresourcetype(1CL), scha_calls(3HA), clconfiguration(5CL), `rbac`(5), r_properties(5)

Notes

The superuser user can run all forms of this command.

Any user can run this command with the following options:

- `-?` option
- `-V` option

To run this command with subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`disable`	`solaris.cluster.admin`
`enable`	`solaris.cluster.admin`

Subcommand	RBAC Authorization
export	solaris.cluster.read
list	solaris.cluster.read
list-props	solaris.cluster.read
monitor	solaris.cluster.admin
reset	solaris.cluster.admin
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

Name clresource, clrs – manage resources for Sun Cluster data services

Synopsis

```
/usr/cluster/bin/clresource subcommand
 [-?]

/usr/cluster/bin/clresource  -V

/usr/cluster/bin/clresource subcommand
 [options] -v  [resource]...

/usr/cluster/bin/clresource clear  [-f errorflag]
 [-g  [resourcegroup,]...] [-t  [resourcetype,]...]
 [ [-z zone]  -n node[:zone][,...]] {+ |  resource...}

/usr/cluster/bin/clresource create  -g resourcegroup
 -t resourcetype [-d] [-p  "property_name[{node_specifier}]"=value]
 [-x  "extension_property[{node_specifier}]"=value]
 [-y  standard_property=value] resource

/usr/cluster/bin/clresource create  -i {- |  clconfiguration}
 -t resourcetype [-a] [-d] [-g  [resourcegroup,]...]
 [-p  "property_name[{node_specifier}]"=value] [-x
 "extension_property[{node_specifier}]"=value] [-y
 standard_property=value] {+ |  resource...}

/usr/cluster/bin/clresource delete  [-F] [-g  [resourcegroup,]...]
 [-t  [resourcetype,]...] {+ |  resource...}

/usr/cluster/bin/clresource disable  [-r] [-g  [resourcegroup,]...]
 [-t  [resourcetype,]...] [ [-z zone]  -n node[:zone][,...]]
 {+ |  resource...}

/usr/cluster/bin/clresource enable  [-r] [-g  [resourcegroup,]...]
 [-t  [resourcetype,]...] [ [-z zone]  -n node[:zone][,...]]
 {+ |  resource...}

/usr/cluster/bin/clresource export  [-o  {- |  configfile}]
 [+ |  resource...]

/usr/cluster/bin/clresource list  [-g  [resourcegroup,]...]
 [-t  [resourcetype,]...] [ [ [-z zone]  -n node[:zone][,...]]]
 [+ |  resource...]

/usr/cluster/bin/clresource list-props  [-l listtype]
 [-g  [resourcegroup,]...] [-p  "property_name[{node_specifier,...}]",...]
 [-t  [resourcetype,]...] [-x  "extension_property[{node_specifier,...}]"...]
 [-y  "standard_property[{node_specifier,...}]",...]
 [+ |  resource...]

/usr/cluster/bin/clresource monitor  [-g  [resourcegroup,]...]
 [-t  [resourcetype,]...] [ [-z zone]  -n node[:zone][,...]]
 {+ |  resource...}

/usr/cluster/bin/clresource set  [-g  [resourcegroup,]...]
 [-p "property_name[{node_specifier,...}]" [+=
```

```
 |  -=]value] [-t  [resourcetype,]...] [-x "extension_property[{node_specifier,...}]"
 [+= |  -=]value] [-y standard_property [+= |  -=]value]
 {+ |  resource...}

/usr/cluster/bin/clresource show  [-g  [resourcegroup,]...]
 [-p  property_name[{node_specifier,...}]",...]
 [-t  [resourcetype,]...] [-x  "extension_property[{node_specifier,...}]",...]
 [-y  "standard_property[{node_specifier,...}]",...]
 [+ |  resource...]

/usr/cluster/bin/clresource status  [-g  [resourcegroup,]...]
 [-s  [state,]...] [-t  [resourcetype,]...] [ [-z zone]
 -n node[:zone][,...]] [+ |  resource...]

/usr/cluster/bin/clresource unmonitor  [-g  [resourcegroup,]...]
 [-t  [resourcetype,]...] [ [-z zone]  -n node[:zone][,...]]
 {+ |  resource...}
```

Description

The `clresource` command manages resources for Sun Cluster data services. The `clrs` command is the short form of the `clresource` command. The `clresource` command and the `clrs` command are identical. You can use either form of the command.

The general form of this command is as follows:

```
clresource  [subcommand] [options] [operands]
```

You can omit *subcommand* only if *options* specifies the `-?` option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are provided with the description of the option in the OPTIONS section of this man page.

Operation in Zones

You can use the `clresource` command with all subcommands in the global zone. When in a non-global zone, the following restrictions apply:

- The `export`, `list`, `list-props`, `show`, and `status` subcommands are unrestricted.
- The `clear`, `disable`, `enable`, `monitor`, `set`, and `unmonitor` subcommands can operate on resources that can be mastered by the non-global zone.
- The `create` and `delete` subcommands can operate on resources that can be mastered by the non-global zone and that have the property `Global_zone=FALSE`.

For ease of administration, you might prefer to run all forms of the command in the global zone.

Resource State and Status

The resource state and resource status are maintained on a per-node basis. A given resource has a distinct state on each cluster node and a distinct status on each cluster node.

Resource Group Manager (RGM) sets the resource state on each node, based on which methods have been invoked on the resource. For example, after the `STOP` method has run successfully on a resource on a given node, the resource's state will be `OFFLINE` on that node. If the `STOP` method exits nonzero or times out, then the state of the resource is `Stop_failed`.

Possible resource states include the following:

- `Online`
- `Offline`
- `Start_failed`
- `Stop_failed`
- `Monitor_failed`
- `Online_not_monitored`
- `Starting`
- `Stopping`
- `Not_online`

Note – State names, such as `Offline` and `Start_failed`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify state names.

In addition to resource state, the RGM also maintains a resource status that can be set by the resource itself by using the API. The field `Status Message` actually consists of two components: status keyword and status message. Status message is optionally set by the resource and is an arbitrary text string that is printed after the status keyword.

Descriptions of possible values for a resource's status are as follows:

`DEGRADED` The resource is online, but its performance or availability might be compromised in some way.

`FAULTED` The resource has encountered an error that prevents it from functioning.

`OFFLINE` The resource is offline.

`ONLINE` The resource is online and providing service.

`UNKNOWN` The current status is unknown or is in transition.

Subcommands The following subcommands are supported:

`clear`
Clears an error flag that is associated with the resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that the error flag is cleared for all resources.

The following options filter the list of operands to limit the resources on which an error flag is cleared:

`-g` *resourcegroup* Clears only the resources in the list of operands that are members of the resource groups in *resourcegroup*.

`-n` *node* Clears the resources on the specified node or nodes. If you do not provide an `-n` option, the command clears resources on all nodes.

`-t` *resourcetype* — Clears only the resources in the list of operands that are instances of the resource types in *resourcetype*.

When in a non-global zone, `clresource clear` operates on resources that can be mastered by the non-global zone.

By default, the `clear` subcommand clears the `STOP_FAILED` error flag. To specify explicitly the error flag that is to be cleared, use the `-f` option. The only error flag that the `-f` option accepts is the `STOP_FAILED` error flag.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

`create`

Creates the resources that are specified as operands to the command.

When you use `create` with the `-i` option to specify a configuration file, the subcommand accepts the plus sign (+) as an operand. When you use the + operand, all resources in the configuration file that do not exist are created.

By default, resources are created in the enabled state with monitoring enabled. However, a resource is brought online and is monitored only after the resource's resource group is brought online. To create resources in the disabled state, specify the `-d` option.

Use the following options to set property values when creating a resource:

`-p` *property_name=value* — Sets standard or extension properties, as long as their names are unique.

`-x` *extension_property=value* — Sets extension properties.

`-y` *standard_property=value* — Sets standard properties.

node_specifier is an *optional* qualifier to the `-p` and `-x` options. It indicates that the properties on *only* the specified node or nodes, or on the specified zone or zones, are to be set when the resource is created. The specified properties on other nodes or zones in the cluster are not set. If you do not include *node_specifier*, the specified properties in all zones on all nodes in the cluster are set. Examples of the syntax of *node_specifier* include the following:

```
-x "myprop{phys-schost-1}"
```

The braces ({}) indicate that you want to set the specified property on only node `phys-schost-1`. For most shells, braces must be quoted.

You can use the following syntax to set a property on two nodes:

```
-x "myprop{phys-schost-1,phys-schost-2}"
```

When in a non-global zone, `clresource create` operates on resources that can be mastered by the non-global zone and that have the property `Global_zone=FALSE`.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand.

See also the description of the `delete` subcommand.

`delete`
Deletes the resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are deleted.

This subcommand deletes multiple resources in the order that is required to satisfy dependencies between the resources. The subcommand disregards the order in which you specify resources on the command line.

When you delete multiple resources at the same time, the command is carried out in several steps. If the command is interrupted, for example, if a node fails, some resources might be left in an invalid configuration. To correct the problem and finish deleting the resources, reissue the same command on a healthy node.

The following options filter the list of operands to limit the resources that are deleted:

`-g` *resourcegroup*	Deletes only the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-t` *resourcetype*	Deletes only the resources in the list of operands that are instances of the resource types in *resourcetype*.

By default, a resource is deleted *only* if the following conditions are met:

- The resource must be disabled.
- All dependencies on the resource must be eliminated.

To force deletion of the specified resources, specify the `-F` option. Use this option with caution, because it has the following effects:

- All specified resources are deleted, even resources that are not disabled.
- All specified resources are removed from resource-dependency settings of other resources.

These effects might cause a loss of service in the cluster. Dependent resources that are not deleted might also be left in an invalid state or in an error state.

When in a non-global zone, `clresource delete` operates on resources that can be mastered by the non-global zone and that have the property `Global_zone=FALSE`.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

See also the description of the `create` subcommand.

`disable`
Disables the resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are disabled.

The following options filter the list of operands to limit the resources that are disabled:

`-g` *resourcegroup*	Disables only the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-n` *node*	You can use `-n` *node* or `-n` *node*:*zone* to disable resources on one or more nodes or in one or more zones.
`-t` *resourcetype*	Disables only the resources in the list of operands that are instances of the resource types in *resourcetype*.

The `-r` option disables any resources that depend on the resources that are specified as operands to the command. These resources are disabled even if the resources are not specified as operands to the command. The `-g` option and the `-t` option do not apply to resources that are to be disabled solely to satisfy resource dependencies.

This subcommand does not affect the monitoring status of the resource. If the resource was monitored when enabled, it is still monitored after the disable. If the resource is subsequently re-enabled, the resource is also monitored.

This subcommand disables resources in the order that is required to satisfy dependencies between the resources. The subcommand disregards the order in which resources are specified at the command line.

When in a non-global zone, `clresource disable` operates on resources that can be mastered by the non-global zone.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `enable` subcommand.

`enable`
Enables the resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are enabled.

The following options filter the list of operands to limit the resources that are enabled:

`-g` *resourcegroup*	Enables only the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-n` *node*	You can use `-n` *node* or `-n` *node*:*zone* to enable resources on one or more nodes or in one or more zones.

`-t` *resourcetype* — Enables only the resources in the list of operands that are instances of the resource types in *resourcetype*.

To ensure that all required resource dependencies are satisfied, specify the `-r` option. The `-r` option enables any resources on which the resources that are specified as operands to the command depend. These resources are enabled, even if the resources are not specified as operands to the command. The `-g` option and the `-t` option do not apply to resources that are to be enabled solely to satisfy resource dependencies.

Resources are enabled in the order that is required to satisfy dependencies between the resources. The subcommand disregards the order in which resources are specified at the command line.

When in a non-global zone, `clresource enable` operates on resources that can be mastered by the non-global zone.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `disable` subcommand.

`export`

Exports the cluster resource configuration in the format that is described by the clconfiguration(5CL) man page.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list`

Displays a list of the resources that are specified as operands to the command. By default, all resources are displayed.

The following options filter the list of operands to limit the resources that are displayed:

`-g` *resourcegroup* — Displays only the resources in the list of operands that are members of the resource groups in *resourcegroup*.

`-n` *node* — You can use `-n` *node* or `-n` *node*`:`*zone* to list only those resources that are online in one or more zones or on one or more nodes.

`-t` *resourcetype* — Displays only the resources that are instances of the resource types in *resourcetype*.

This subcommand accepts the plus sign (+) as an operand to specify that all the resource configuration is displayed. You can restrict the displayed information to specific resource

groups or resource types by specifying a `-g` option or `-t` option. If no operands are supplied, all resources in the specified resource groups or that are instances of the specified resource types are displayed.

If you specify the `-v` option, the resource group and resource type of each resource in the list are also displayed.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list-props`
Displays a list of the properties of the resources that are specified as operands to the command.

The following options filter the list of operands to limit the resources whose properties are displayed:

`-g` *resourcegroup*	Displays the properties only of the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-t` *resourcetype*	Displays the properties only of the resources in the list of operands that are instances of the resource types in *resourcetype*.

The `-l` option specifies the type of resource properties that are to be displayed:

`-l all`	Specifies that standard properties and extension properties are displayed.
`-l extension`	Specifies that only extension properties are displayed. By default, only extension properties are displayed.
`-l standard`	Specifies that only standard properties are displayed.

If you do not specify the `-l` option, only extension properties are displayed. To display standard properties, specify the properties explicitly by using the `-p` option or the `-y` option.

The following options limit the set of resource properties that is to be displayed:

`-p` *property_name*	Displays only the properties that are specified in *property_name*. You can specify standard properties and extension properties in *property_name*.
`-x` *extension_property*	Displays only the extension properties on one or more nodes that are specified in *extension_property*.
`-y` *standard_property*	Displays only the standard properties that are specified in *standard_property*.

node_specifier is an *optional* qualifier to the `-p`, `-x`, and `-y` options. It indicates that the properties on *only* the specified node or nodes, or on the specified zone or zones, are to be displayed. The specified properties on other nodes or zones in the cluster are not displayed. If you do not include *node_specifier*, the specified properties in all zones on all nodes in the cluster are displayed. Examples of the syntax of *node_specifier* include the following:

```
-x "myprop{phys-schost-1}"
```

The braces ({}) indicate that you want to display the specified property on only node `phys-schost-1`. For most shells, braces must be quoted.

You can use the following syntax to display a property on two nodes:

```
-x "myprop{phys-schost-1,phys-schost-2}"
```

If you specify the `-v` option, the description of each property is also displayed.

This subcommand accepts the plus sign (+) as an operand to specify that all resource properties are displayed. If no operands are supplied, properties of all resources in the specified resource groups or that are instances of the specified resource types are displayed.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`monitor`

Turns on monitoring for the resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that monitoring is turned on for all resources.

The following options filter the list of operands to limit the resources for which monitoring is turned on:

`-g` *resourcegroup*	Turns on monitoring only for the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-n` *node*	Turns on monitoring for only those resources that are online in one or more zones or on one or more nodes.
`-t` *resourcetype*	Turns on monitoring only for the resources in the list of operands that are instances of the resource types in *resourcetype*.

If monitoring is turned on for a resource, the resource is monitored only if the following conditions are met:

- The resource is enabled.
- The resource group that contains the resource is online on at least one cluster node.

Note – When you turn on monitoring for a resource, you do *not* enable the resource.

When in a non-global zone, `clresource monitor` operates on resources that can be mastered by the non-global zone.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `unmonitor` subcommand.

`set`

Sets specified properties of the resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that the specified properties of all resources are modified.

The following options filter the list of operands to limit the resources for which properties are modified:

`-g` *resourcegroup*	Modifies properties of only the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-t` *resourcetype*	Modifies properties of only the resources in the list of operands that are instances of the resource types in *resourcetype*.

Use the following options to set property values:

`-p` *property_name=value*	Sets standard or extension properties, as long as their names are unique.
`-x` *extension_property=value*	Sets extension properties.
`-y` *standard_property=value*	Sets standard properties.

node_specifier is an *optional* qualifier to the `-p` and `-x` options. It indicates that the properties on *only* the specified node or nodes, or on the specified zone or zones, are to be set when the resource is created. The specified properties on other nodes or zones in the cluster are not set. If you do not include *node_specifier*, the specified properties in all zones on all nodes in the cluster are set. Examples of the syntax of *node_specifier* include the following:

```
-x "myprop{phys-schost-1}"
```

The braces ({}) indicate that you want to set the specified property on only node `phys-schost-1`. For most shells, braces must be quoted.

You can use the following syntax to set a property on two nodes:

```
-x "myprop{phys-schost-1,phys-schost-2}"
```

When in a non-global zone, `clresource set` operates on resources that can be mastered by the non-global zone.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`show`

Displays the configuration of the resources that are specified as operands to the command. By default, the configuration of all resources is displayed.

The following options filter the list of operands to limit the list of resources for which the configuration is displayed:

`-g` *resourcegroup*	Displays the configuration of only the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-n` *node*	You can use `-n` *node* or `-n` *node*`:`*zone* to display the configuration of only those resources that are online in one or more zones or on one or more nodes.
`-t` *resourcetype*	Displays the configuration of only the resources in the list of operands that are instances of the resource types in *resourcetype*.

The following options limit the set of resource properties that are displayed:

`-p` *property_name*	Displays only the properties that are specified in *property_name*. You can specify standard properties and extension properties in *property_name*.
`-x` *extension_property*	Displays only the extension properties on one or more nodes that are specified in *extension_property*.
`-y` *standard_property*	Displays only the standard properties that are specified in *standard_property*.

node_specifier is an *optional* qualifier to the `-p`, `-x`, and `-y` options. It indicates that the properties on *only* the specified node or nodes, or on the specified zone or zones, are to be displayed. The specified properties on other nodes or zones in the cluster are not displayed. If you do not include *node_specifier*, the specified properties in all zones on all nodes in the cluster are displayed. Examples of the syntax of *node_specifier* include the following:

```
-x "myprop{phys-schost-1}"
```

The braces ({}) indicate that you want to display the specified property on only node `phys-schost-1`. For most shells, braces must be quoted.

You can use the following syntax to display a property on two nodes:

```
-x "myprop{phys-schost-1,phys-schost-2}"
```

This subcommand accepts the plus sign (+) as an operand to specify all resource configuration is to be displayed. You can restrict the displayed information to specific resource groups or resource types by specifying a `-g` option or `-t` option. If you do not supply an operand, the subcommand displays the configuration of all specified resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`status`
Displays the status of the resources that are specified as operands to the command. By default, the status of all resources is displayed.

The following options filter the list of operands to limit the list of resources for which the status is displayed:

`-g` *resourcegroup*	Displays the status of only the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-n` *node*	You can use `-n` *node* or `-n` *node*`:`*zone* to display the status of only those resources that are online in one or more zones or on one or more nodes. You cannot specify the `-n` and `-s` options together.
`-s` *state*	Displays the status of only the resources in the list of operands that are in the states in *state*. You cannot specify the `-n` and `-s` options together.
`-t` *resourcetype*	Displays the status of only the resources in the list of operands that are instances of the resource types in *resourcetype*.

This subcommand accepts the plus sign (+) as an operand to specify that the status of all resources is to be displayed. You can restrict the displayed information to specific resource groups or resource types by specifying a `-g` option or `-t` option. If no operands are supplied, the status of all specified resources is displayed.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`unmonitor`
Turns off monitoring for the resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that monitoring is turned off for all resources.

If you turn off monitoring for a resource that is disabled, the resource is not affected. The resource and its monitor are already offline.

Note – When you turn off monitoring for a resource, you do *not* disable the resource. However, when you disable a resource, you do not need to turn off monitoring for the resource. The disabled resource and its monitor are kept offline.

The following options filter the list of operands to limit the resources for which monitoring is turned off:

`-g` *resourcegroup*	Turns off monitoring only for the resources in the list of operands that are members of the resource groups in *resourcegroup*.
`-n` *node*	Turns off monitoring for only those resources that are online in one or more zones or on one or more nodes.
`-t` *resourcetype*	Turns off monitoring only for the resources in the list of operands that are instances of the resource types in *resourcetype*.

When in a non-global zone, `clresource unmonitor` operates on resources that can be mastered by the non-global zone.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `disable` subcommand and the `monitor` subcommand.

Options The following options are supported:

`-?`
`--help`
Displays help information. When this option is used, no other processing is performed.

You can specify this option without a subcommand or with a subcommand.

If you specify this option without a subcommand, the list of subcommands for this command is displayed.

If you specify this option with a subcommand, the usage options for the subcommand are displayed.

`-a`
`--automatic`
Automatically performs the following additional operations when resources are being created from a cluster configuration file (clconfiguration(5CL)):

- Registering resource types
- Creating resource groups
- Creating resources on which the resources that are specified in the list of operands depend

The cluster configuration information must contain sufficient information to do all of the following:

- Enable the resource types to be registered
- Enable the resource groups to be created
- Enable the resources to be created

You can specify this option only with the `create` subcommand. If you specify this option, you must also specify the `-i` option and provide a configuration file.

`-d`
`--disable`
Disables a resource when the resource is created. You can specify this option only with the `create` subcommand. By default, resources are created in the enabled state.

Enabling a resource does not guarantee that the resource is brought online. A resource is brought online only after the resource's resource group is brought online on at least one node.

`-f` *errorflag*
`--flag=`*errorflag*
`--flag` *errorflag*
Specifies explicitly the error flag that is to be cleared by the `clear` subcommand. You can specify this option only with the `clear` subcommand. By default, the `clear` subcommand clears the `STOP_FAILED` error flag.

The only error flag that the `-f` option accepts is the `STOP_FAILED` error flag.

`-F`
`--force`
Forces the deletion of resources that are not disabled. You can specify this option only with the `delete` subcommand.

Use this option with caution, because it has the following effects:

- All specified resources are deleted, even resources that are not disabled.
- All specified resources are removed from resource-dependency settings of other resources.

These effects might cause a loss of service in the cluster. Dependent resources that are not deleted might also be left in an invalid state or in an error state.

`-g` *resourcegroup*[,…]
`--resourcegroup=`*resourcegroup*`-[,…]`
`--resourcegroup` *resourcegroup*`-[,…]`
Specifies a resource group or a list of resource groups.

For subcommands except `create`, the command acts on only the resources in the list of operands that are members of the specified resource groups. Specify resource groups by using the `-g` option.

When you specify the `-g` option with the `create` subcommand, `clresource` creates the resource in the specified resource group. You can specify only one resource group when using this option.

`-i` {`-` | *clconfiguration*}
`--input=`{`-` | *clconfiguration*`-`}
`--input` {`-` | *clconfiguration*`-`}
Specifies configuration information that is to be used for creating or modifying resources. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through the standard input. To specify the standard input, specify `-` instead of a file name.

Only the resources that are supplied as operands to the command are created or are modified. Options that are specified in the command override any options that are set in the configuration information. If configuration parameters are missing in the configuration information, you must specify these parameters at the command line.

When you use the `-i` option with the `create` subcommand, `clresource` registers all required resource types and creates all required resource groups. You must supply all information that is required for the registration and configuration. All other configuration data is ignored.

`-l` *listtype*
`--listtype=`*listtype*
`--listtype` *listtype*
Specifies the type of resource properties that are to be displayed by the `list-props` subcommand. You can specify this option only with the `list-props` subcommand.

You must specify one value from the following list for *listtype*:

`all`	Specifies that standard properties and extension properties are displayed.
`extension`	Specifies displayed only extension properties are displayed. By default, only extension properties are displayed.
`standard`	Specifies that only standard properties are displayed.

If you do not specify the `-l` option, only extension properties are displayed. To display standard properties, specify the properties explicitly by using the `-p` option or the `-y` option.

`-n` *node*`-[:`*zone*`-][,…]`
`--node=`*node*`-[:`*zone*`-][,…]`
`--node` *node*`-[:`*zone*`-][,…]`
Specifies a node or a list of nodes. You can specify each node as a node name or a node ID. For each node, you can optionally specify a non-global zone on the node.

The subcommands with which you can specify this option are as follows:

`disable`	Disables only the resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`enable`	Enables only the resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`list`	Displays a list of only those resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`monitor`	Monitors only those resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`show`	Displays the configuration information of only those resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`status`	Reports the status only of resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`unmonitor`	Unmonitors only those resources in the list of operands that are hosted on the specified nodes or in the specified zones.

`-o {- | `*clconfiguration*`-}`
`--output={- | `*clconfiguration*`-}`
`--output {- | `*clconfiguration*`-}`
Specifies the location where resource configuration information is to be written. This location can be a file or the standard output. To specify the standard output, specify a dash (`-`) instead of a file name. If you specify the standard output, all other standard output for the command is suppressed. You can specify this option only with the `export` subcommand.

Configuration information is written only for the resources that are supplied as operands to the command. The information is written in the format that is defined in the clconfiguration(5CL) man page.

`-p` *property_name*`-=`*value*
`-p` *property_name*`-+=`*array-values*
`-p` *property_name*`--=`*array-values*
`--property=`*property_name*`-=`*value*
`--property` *property_name*`-=`*value*
`--property=`*property_name*`-+=`*array-values*
`--property` *property_name*`-+=`*array-values*
`--property=`*property_name*`--=`*array-values*
`--property` *property_name*`--=`*array-values*
Sets the values of a property for resources that are supplied as operands to the command. You can specify the assignment form of this option only with the `create` subcommand and the `set` subcommand.

Use the `-p` option to specify any standard or extension property. If an extension property of a resource type has the same name as a standard property of that resource type, use of the `-p` option returns an error. In this situation, use the `-x` option to specify the extension property and the `-y` option to specify the standard property.

For a description of standard properties, see the r_properties(5) man page.

For a description of a resource type's extension properties, see the documentation for the resource type.

The operators to use with this option are as follows:

`=` Sets the property to the specified value. The `create` subcommand and the `set` subcommand accept this operator.

`+=` Adds a value or values to a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for properties that accept lists of string values, for example `Node`.

`-=` Deletes a value or values from a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for properties that accept lists of string values, for example `Node`.

To set a per-node property on a subset of cluster nodes or zones, specify the nodes or zones when the property is set. Append the list of nodes or zones in braces to the property name as follows:

name{*node[:zone]*}

node is a comma-separated list of node names, node IDs, or zones. For more information about per-node properties, see the rt_properties(5) man page.

`-p` *property_name*`-[,…]`
`--property=`*property_name*`-[,...]`
`--property` *property_name*`-[,...]`
Specifies a list of properties for the `list-props` subcommand and `show` subcommand.

Use the `-p` option to specify any standard or extension property. If an extension property of a resource type has the same name as a standard property of that resource type, use of the `-p` option returns an error. In this situation, use the `-x` option to specify the extension property and the `-y` option to specify the standard property.

For a description of standard properties, see the r_properties(5) man page.

For a description of a resource type's extension properties, see the documentation for the resource type.

Without this option, the `list-props` subcommand and `show` subcommand list all or most resource properties, depending on whether the `-v` option is also specified.

`-r`
`--recursive`
Recursively enables or disables resources to ensure that all required dependencies are satisfied. You can specify this option only with the `disable` subcommand and the `enable` subcommand.

The effect of this option with these subcommands is as follows:

`disable`	Disables any resources that depend on the resources that are specified as operands to the command. The resources are disabled even if the resources are not specified as operands to the command.
`enable`	Enables any resources on which resources that are specified as operands to the command depend. The resources are enabled even if the resources are not specified as operands to the command.

`-s` *state*`-[,…]`
`--state=`*state*`-[,...]`
`--state` *state*`-[,...]`
Specifies a list of states for the `list` subcommand and `status` subcommand.

This option limits the output to include only those resources that are in one of the specified states on one or more nodes in the node list.

The possible states are as follows:

- `Online`
- `Offline`
- `Start_failed`
- `Stop_failed`
- `Monitor_failed`
- `Online_not_monitored`
- `Starting`
- `Stopping`
- `Not_online`

Note – State names, such as `Offline` and `Start_failed`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify state names.

`-t` *resourcetype*`-[,…]`
`--type=`*resourcetype*`-[,...]`
`--type` *resourcetype*`-[,...]`
Specifies a resource type or list of resource types.

For all subcommands that accept this option except `create`, the command acts only on resources that satisfy both of the following qualifications:

- The resources are in the list of operands.
- The resources are instances of the resource types that the `-t` option specifies.

When you specify the `-t` option with `clresource create`, you create a resource of the specified type. You can specify only one resource type.

For a description of the format of resource type names, see "RGM Legal Names" in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS* in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*.

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The `-V` option displays only the version of the command. No other operations are performed.

`-v`
`--verbose`
Displays verbose messages to the standard output.

You can specify this option with any form of the command.

Do not specify the `-v` option with the `-o -` option. The `-v` option is ignored. The `-o -` option suppresses all other standard output.

`-x` *extension_property*`-=`*value*
`-x` *extension_property*`-+=`*array-value*
`-x` *extension_property*`--=`*array-value*
`--extension-property=`*extension_property*`-=`*value*
`--extension-property` *extension_property*`-=`*value*
`--extension-property=`*extension_property*`-+=`*array-value*
`--extension-property` *extension_property*`-+=`*array-value*
`--extension-property=`*extension_property*`--=`*array-value*
`--extension-property` *extension_property*`--=`*array-value*
Sets or modifies the value of an extension property of resources that are supplied as operands to the command.

In general, use the `-p` option to specify any standard or extension property. If an extension property of a resource type has the same name as a standard property of that resource type, use of the `-p` option returns an error. In this situation, use the `-x` option to specify the extension property and the `-y` option to specify the standard property.

You can specify the assignment form of this option only with the `create` subcommand and the `set` subcommand.

For a description of a resource type's extension properties, see the documentation for the resource type.

The operators to use with this option are as follows:

`=` Sets the property to the specified value. The `create` subcommand and the `set` subcommand accept this operator.

`+=` Adds a value or values to a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for properties that accept lists of string values, for example `Node`.

`-=` Removes a value or values from a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for properties that accept lists of string values, for example `Node`.

To set a per-node property on a subset of cluster nodes or zones, specify the nodes or zones when the property is set. Append the list of nodes or zones in braces to the property name as follows:

name`{`*node*`[:`*zone*`]}`

node is a comma-separated list of node names, node IDs, or zones. For more information about per-node properties, see the rt_properties(5) man page.

`-x` *extension_property*`-[,…]`
`--extension-property=`*extension_property*`-[,...]`
`--extension-property` *name*`-[,...]`
Specifies a list of extension properties for the `list-props` subcommand and the `show` subcommand.

For a description of a resource type's extension properties, see the documentation for the resource type.

Use the `-p` option to specify any standard or extension property. If an extension property of a resource type has the same name as a standard property of that resource type, use of the `-p` option returns an error. In this situation, use the `-x` option to specify the extension property and the `-y` option to specify the standard property.

Without this option, the `list-props` subcommand and the `show` subcommand list all or most resource properties, depending on whether the `-v` option is also specified.

`-y` *standard_property*`-=`*value*
`-y` *standard_property*`-+=`*array-value*
`-y` *standard_property*`--=`*array-value*
`--standard-property=`*standard_property*`-=`*value*
`--standard-property` *standard_property*`-=`*value*
`--standard-property=`*standard_property*`-+=`*array-value*
`--standard-property` *standard_property*`-+=`*array-value*
`--standard-property=`*standard_property*`--=`*array-value*
`--standard-property` *standard_property*`--=`*array-value*
Sets or modifies the value of a standard property of resources that are supplied as operands to the command.

Use the `-p` option to specify any standard or extension property. If an extension property of a resource type has the same name as a standard property of that resource type, use of the `-p` option returns an error. In this situation, use the `-x` option to specify the extension property and the `-y` option to specify the standard property.

You can specify the assignment form of this option only with the `create` subcommand and the `set` subcommand.

For a description of standard properties, see the r_properties(5) man page.

The operators to use with this option are as follows:

`=` Sets the property to the specified value. The `create` subcommand and the `set` subcommand accept this operator.

`+=` Adds a value or values to a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for properties that accept lists of string values, for example `Node`.

`-=` Removes a value or values from a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for properties that accept lists of string values, for example `Node`.

To set a per-node property on a subset of cluster nodes, specify the subset nodes. Append the list of nodes in braces to the property name as follows:

"*standard_property*{*node_specifier*}"

node_specifier is a comma-separated list of node names, node IDs, or zones. It indicates that only the properties on the node or nodes, or on the zone or zones, are affected by the command. For more information about per-node properties, see the rt_properties(5) man page.

`-y` *standard_property*`-[,…]`
`--standard-property=`*standard_property*`-[,...]`
`--standard-property` *standard_property*`-[,...]`
Specifies a list of standard properties for the `list-props` subcommand and `show` subcommand.

For a description of standard properties, see the r_properties(5) man page.

Use the `-p` option to specify any standard or extension property. If an extension property of a resource type has the same name as a standard property of that resource type, use of the `-p` option returns an error. In this situation, use the `-x` option to specify the extension property and the `-y` option to specify the standard property.

Without this option, the `list-props` subcommand and the `show` subcommand list all or most resource properties, depending on whether the `-v` option is also specified.

`-z` *zone*
`--zone=`*zone*
`--zone` *zone*
Applies the same zone name to all nodes in a node list for which a zone is not explicitly specified. You can specify this option only with the `-n` option.

Operands Only the following operand is supported:

resource Specifies the resource that is to be managed or the resources that are to be managed. If the subcommand accepts more than one resource, you can use the plus sign (+) to specify all resources.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

The following exit codes can be returned:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`9` `CL_ESTATE`
Object is in wrong state

You tried to modify a property, a resource group, or other object that you cannot modify at that particular time or at any time.

`10` `CL_EMETHOD`
Resource method failed

A method of a resource failed. The method failed for one of the following reasons:

- The `validate` method failed when you tried to create a resource or modify the properties of a resource.
- A method other than `validate` failed when you tried to enable, disable, or delete a resource.

`15 CL_EPROP`
Invalid property

The property or value that you specified with the `-p`, `-y`, or `-x` option does not exist or is not allowed.

`36 CL_ENOENT`
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

`39 CL_EEXIST`
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

`41 CL_ETYPE`
Invalid type

The type that you specified with the `-t` or `-p` option does not exist.

These exit values are compatible with the return codes that are described in the scha_calls(3HA) man page.

Examples

EXAMPLE 1 Creating a Resource

This example creates a resource that is named `rs-nfs` in a resource group that is named `rg-failover`. The resource is an instance of the `SUNW.nfs` resource type. The resource is created in the enabled state and with resource monitoring turned on.

```
# clresource create -g rg-failover -t SUNW.nfs rs-nfs
```

EXAMPLE 2 Turning On Monitoring for a Resource

This example turns on monitoring for a resource that is named `rs-nfs`.

```
# clresource monitor rs-nfs
```

EXAMPLE 2 Turning On Monitoring for a Resource *(Continued)*

When monitoring is turned on for a resource, it remains on until explicitly turned off by using the `clresource unmonitor` command. Disabling and enabling a resource does not affect whether it is monitored.

EXAMPLE 3 Enabling Resources

This example enables all resources in resource groups `rg-failover` and `rg-failover2`.

```
# clresource enable -g rg-failover,rg-failover2 +
```

This command does not affect whether the resources are monitored.

EXAMPLE 4 Setting a Resource Property

This example sets the `r_description` property of all instances of the `SUNW.nfs` resource type to `HA-NFS res`.

```
# clresource set -t SUNW.nfs -p r_description="HA-NFS res" +
```

EXAMPLE 5 Setting a Per-Node Resource Property

This example sets the per-node property `oracle_sid` of the resource `rs-oracle` to different values on different nodes, as follows:

- On node `phys-schost-1` and node `phys-schost-2`, this property is set to `myora1`.
- On node `phys-schost-3`, this property is set to `myora2`.

This example assumes that the brace character has a special meaning to the shell that is used. Therefore, each property name to which the node list is appended is enclosed in double quotes.

```
# clresource set -p "oracle_sid{phys-schost-1,phys-schost-2}"=myora1 \\
-p "oracle_sid{phys-schost-3}"=myora2 rs-oracle
```

EXAMPLE 6 Adding a Value to a String-Array Property

This example adds the value `rs-oracle` to the string-array property `resource_dependencies` of the resource `rs-myapp`. Existing values in this string-array property are unchanged.

```
# clresource set -p resource_dependencies+=rs-oracle rs-myapp

# clresource show -p resource_dependencies rs-myapp
Resource: rs-myapp
Standard Properties:
Resource_dependencies: rs-nfs rs-oracle
```

EXAMPLE 7 Deleting a Resource

This example deletes a resource that is named `rs-nfs`.

```
# clresource delete rs-nfs
```

EXAMPLE 8 Updating an Entire Cluster Configuration

This example updates an entire cluster configuration by performing the following sequence of operations:

1. Bringing offline all resource groups in the cluster, deleting all resources, and deleting all resource groups
2. Unregistering all resource types
3. Creating all resources that are specified in the configuration file `/net/server/export/mycluster.xml`, registering their resource types, and creating all required resource groups

```
# clresourcegroup delete --force +
# clresourcetype unregister +
# clresource -i /net/server/export/mycluster.xml -a +
```

EXAMPLE 9 Listing Resources

This example lists all resources.

```
# clresource list
logicalhost1
rs-nfs-1
rs-nfs-2
logicalhost2
rs-apache-1
```

EXAMPLE 10 Listing Resources With Groups and Types

This example lists all resources with their resource groups and resource types.

```
# clresource list -v

Resource Name      Resource Group    Resource Type
-------------      --------------    -------------
logicalhost1       rg-failover-1     SUNW.LogicalHostname
rs-nfs-1           rg-failover-1     SUNW.nfs
logicalhost2       rg-failover-2     SUNW.LogicalHostname
rs-nfs-2           rg-failover-2     SUNW.nfs
rs-apache-1        rg-failover-1     SUNW.apache
```

EXAMPLE 11 Listing Resources of a Specific Type

This example lists all instances of the `nfs` resource type.

```
# clresource list -t nfs
rs-nfs-1
rs-nfs-2
```

EXAMPLE 12 Listing Extension Properties and Descriptions for a Resource Type

This example lists the extension properties and a description of each extension property for the `nfs` resource type.

```
# clresource list-props -t nfs -v
Properties              Descriptions
----------              ------------
Monitor_retry_count     Number of PMF restarts allowed for the fault monitor
Monitor_retry_interval  Time window (minutes) for fault monitor restarts
Rpcbind_nullrpc_timeout Timeout(seconds) to use when probing rpcbind
Nfsd_nullrpc_timeout    Timeout(seconds) to use when probing nfsd
Mountd_nullrpc_timeout  Timeout(seconds) to use when probing mountd
Statd_nullrpc_timeout   Timeout(seconds) to use when probing statd
Lockd_nullrpc_timeout   Timeout(seconds) to use when probing lockd
Rpcbind_nullrpc_reboot  Boolean to indicate if we should reboot system when
                          null rpc call on rpcbind fails
Nfsd_nullrpc_restart    Boolean to indicate if we should restart nfsd when
                          null rpc call fails
Mountd_nullrpc_restart  Boolean to indicate if we should restart mountd when
                          null rpc call fails
```

Line breaks in the Descriptions column are added to enhance the readability of this example. Actual output from the command does not contain these line breaks.

EXAMPLE 13 Clearing a `Start_failed` Resource State by Disabling and Enabling a Resource

The `Start_failed` resource state indicates that a `Start` or `Prenet_start` method failed or timed out on a resource, but its resource group came online anyway. The resource group comes online even though the resource has been placed in a faulted state and might not be providing service. This state can occur if the resource's `Failover_mode` property is set to `None` or to another value that prevents the failover of the resource group.

Unlike the `Stop_failed` resource state, the `Start_failed` resource state does *not* prevent you or the Sun Cluster software from performing actions on the resource group. You do not need to issue the *command* `clear` command to clear a `Start_failed` resource state. You only need to execute a command that restarts the resource.

The following command clears a `Start_failed` resource state that has occurred on the resource `resource-1` by disabling and then re-enabling the resource.

EXAMPLE 13 Clearing a `Start_failed` Resource State by Disabling and Enabling a Resource *(Continued)*

```
# clresource disable resource-1
# clresource enable resource-1
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), clreslogicalhostname(1CL), clresourcegroup(1CL), clresourcetype(1CL), clressharedaddress(1CL), cluster(1CL), scha_calls(3HA), clconfiguration(5CL), `attributes`(5), r_properties(5), `rbac`(5)

Sun Cluster Data Services Planning and Administration Guide for Solaris OS

Notes

The superuser can run all forms of this command.

Any user can run this command with the following options:

- `-?` option
- `-V` option

To run this command with subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`disable`	`solaris.cluster.admin`
`enable`	`solaris.cluster.admin`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`list-props`	`solaris.cluster.read`
`set`	`solaris.cluster.modify`
`monitor`	`solaris.cluster.admin`

Subcommand	RBAC Authorization
clear	solaris.cluster.admin
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

Name clresourcegroup, clrg – manage resource groups for Sun Cluster data services

Synopsis

```
/usr/cluster/bin/clresourcegroup  -V

/usr/cluster/bin/clresourcegroup [subcommand]
 -?

/usr/cluster/bin/clresourcegroup subcommand
 [options] -v [resourcegroup ...]

/usr/cluster/bin/clresourcegroup add-node  -n node[:zone][,...]
 [-S] [-z zone] {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup create  [-S] [ -n
 node[:zone][,...]] [-p name=value] [-z zone] [...]
 resourcegroup...

/usr/cluster/bin/clresourcegroup create  -i {-
 |  clconfigfile} [-S] [ -n node[:zone][,...]] [-p name=value]
 [-z zone] [...] {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup delete  [-F] {+
 |  resourcegroup...}

/usr/cluster/bin/clresourcegroup evacuate  -n  node[:zone][,...]
 [-T seconds] [-z zone] [+]

/usr/cluster/bin/clresourcegroup export  [-o  {-
 |  clconfigfile}] [+ |  resourcegroup...]

/usr/cluster/bin/clresourcegroup list  [-n node[:zone][,...]]
 [-r resource[,...]] [-s state[,...]] [-t resourcetype[,...]]
 [ [-z zone]] [+ |  resourcegroup...]

/usr/cluster/bin/clresourcegroup manage  {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup offline  [-n node[:zone][,...]]
 [-z zone] {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup online  [-e] [-m]
 [-M] [-n node[:zone][,...]] [-z zone] {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup quiesce  [-k] {+
 |  resourcegroup...}

/usr/cluster/bin/clresourcegroup remaster  {+
 |  resourcegroup...}

/usr/cluster/bin/clresourcegroup remove-node  -n node[:zone][,...]
 [-z zone] {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup restart  [-n node[:zone][,...]]
 {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup resume  {+ |  resourcegroup...}
```

```
/usr/cluster/bin/clresourcegroup set  [-i  {- |  clconfigfile}]
 [-n node[:zone][,...]] [-p name[+|-]=value] [...]
 [-z zone] {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup show  [-n node[:zone][,...]]
 [-p name[,...]] [-r resource[,...]] [-t resourcetype[,...]]
 [-z zone] [+ |  resourcegroup...]

/usr/cluster/bin/clresourcegroup status  [-n node[:zone][,...]]
 [-r resource [,]...] [-s state [,]...] [-t resourcetype
 [,]...] [-z zone] [+ |  resourcegroup...]

/usr/cluster/bin/clresourcegroup suspend  [-k] {+
 |  resourcegroup...}

/usr/cluster/bin/clresourcegroup switch  -n node[:zone][,...]
 [-e] [-m] [-M] [-z zone] {+ |  resourcegroup...}

/usr/cluster/bin/clresourcegroup unmanage  {+
 |  resourcegroup...}
```

Description

This command manages Sun Cluster data service resource groups.

You can omit *subcommand* only if *options* is the `-?` option or the `-V` option.

Each option has a long and a short form. Both forms of each option are given with the description of the option in OPTIONS.

The `clrg` command is the short form of the `clresourcegroup` command.

With the exception of `list`, `show`, and `status`, subcommands require at least one operand. But, many subcommands accept the plus sign operand (+). This operand applies the subcommand to *all* applicable objects.

You can use some forms of this command in a non-global zone, referred to simply as a zone. For more information about valid uses of this command in zones, see the descriptions of the individual subcommands. For ease of administration, use this command in the global zone.

Resources and Resource Groups

The resource state, resource group state, and resource status are all maintained on a per-node basis. For example, a given resource has a distinct state on each cluster node and a distinct status on each cluster node.

Note – State names, such as `Offline` and `Start_failed`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify state names.

The resource state is set by the Resource Group Manager (RGM) on each node, based only on which methods have been invoked on the resource. For example, after the `STOP` method has run successfully on a resource on a given node, the resource's state is `Offline` on that node. If the `STOP` method exits nonzero or times out, the state of the resource is `Stop_failed`.

Possible resource states include: `Online`, `Offline`, `Start_failed`, `Stop_failed`, `Monitor_failed`, `Online_not_monitored`, `Starting`, and `Stopping`.

Possible resource group states are: `Unmanaged`, `Online`, `Offline`, `Pending_online`, `Pending_offline`, `Error_stop_failed`, `Online_faulted`, and `Pending_online_blocked`.

In addition to resource state, the RGM also maintains a resource status that can be set by the resource itself by using the API. The field `Status Message` actually consists of two components: status keyword and status message. Status message is optionally set by the resource and is an arbitrary text string that is printed after the status keyword.

Descriptions of possible values for a resource's status are as follows:

`Degraded`	The resource is online, but its performance or availability might be compromised in some way.
`Faulted`	The resource has encountered an error that prevents it from functioning.
`Offline`	The resource is offline.
`Online`	The resource is online and providing service.
`Unknown`	The current status is unknown or is in transition.

Subcommands The following subcommands are supported:

`add-node`
Adds a node or zone to the end of the `Nodelist` property for a resource group.

You can use this subcommand only in the global zone.

The order of the nodes and zones in the list specifies the preferred order in which the resource group is brought online on those nodes or zones. To add a node or zone to a different position in the `Nodelist` property, use the `set` subcommand.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

`create`
Creates a new resource group.

You can use this subcommand only in the global zone.

If you specify a configuration file with the `-i` option, you can specify the plus sign operand (+). This operand specifies that you want to create all resources in that file that do not yet exist.

To set the `Nodelist` property for the new resource group, specify one of the following options:

- `-n` *node*`-[:`*zone*`-][,…]`
- `-p Nodelist=-`*node*`--[:`*zone*`-][,…]`
- `-i` *clconfigfile*

The order of the nodes or zones in the list specifies the preferred order in which the resource group is brought online on those nodes or zones. If you do not specify a node list at creation, the `Nodelist` property is set to all nodes and zones that are configured in the cluster. The order is arbitrary.

By default, resource groups are created with the `RG_mode` property set to `Failover`. However, by using the `-S` option or the `-p RG_mode=Scalable` option, or by setting `Maximum_primaries` to a value that is greater than 1, you can create a scalable resource group. You can set the `RG_mode` property of a resource group only when that group is created.

Resource groups are always placed in an unmanaged state when they are created. However, when you issue the `manage` subcommand, or when you issue the `online` or `switch` subcommand with the `-M` option, the RGM changes their state to a managed state.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`delete`
Deletes a resource group.

You can use this subcommand only in the global zone.

You can specify the plus sign operand (+) with this subcommand to delete all resource groups.

You cannot delete resource groups if they contain resources, unless you specify the `-F` option. If you specify the `-F` option, all resources within each group, as well as the group, are deleted. All dependencies and affinities are deleted as well.

This subcommand deletes multiple resource groups in an order that reflects resource and resource group dependencies. The order in which you specify resource groups on the command line does not matter.

The following forms of the `clresourcegroup delete` command are carried out in several steps:

- When you delete multiple resource groups at the same time
- When you delete a resource group with the `-F` option

If either of these forms of the command is interrupted, for example, if a node fails, some resource groups might be left in an invalid configuration.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`evacuate`
Brings offline all resource groups on the nodes or zones that you specify with the `-n` option.

When you specify a physical node (or global zone), this subcommand evacuates all resource groups, including resource groups in all zones, off the specified node.

When run in a global zone, this subcommand can evacuate any specified node or zone in the cluster. When run in a non-global zone, this subcommand can only evacuate that non-global zone.

Resource groups are brought offline in an order that reflects resource and resource group dependencies. The order in which you specify resource groups on the command line does not matter.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`export`
Writes the configuration information for a resource group to a file or to the standard output (`stdout`).

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resource groups that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resource groups.

The format of this configuration information is described in the clconfiguration(5CL) man page.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`list`
Displays a list, filtered by qualifier options, of resource groups that you specify.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resource groups that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resource groups.

You can use `-r` *resource* to include only those resource groups that contain resources. You can use `-t` *resourcetype* to include only those resource groups that contain a resource type in *resourcetype*. You can use `-n` *node* or `-n` *node:zone* to include only those resource groups that are online in one or more zones or on one or more nodes.

If you specify `-s` *state*, only those groups with the states that you specify are listed.

If you do not specify an operand or if you specify the plus sign operand (+), all resource groups, filtered by any qualifier options that you specify, are listed.

If you specify the verbose option `-v`, the status (whether the resource group is online or offline) is displayed. A resource group is listed as online even if it is online on only one node or zone in the cluster.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`manage`
Brings a resource group that you specify to a managed state.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`offline`
Brings a resource group that you specify to an offline state.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

If you specify the `-n` option, only resource groups on the nodes or zones that you specify are taken offline.

If you do not specify the `-n` option, resource groups on all nodes and zones are brought offline.

Resource groups are brought offline in an order that reflects resource and resource group dependencies. The order in which you specify groups on the command line does not matter.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`online`
Brings a resource group that you specify to an online state.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

Use the `-n` option to specify the list of nodes or zones on which to bring resource groups online. If you do not specify the `-n` option and no resource-group affinities exist, this subcommand brings nodes and zones online in the order that is specified by the `Nodelist` property. For failover groups, this node or zone is the first online node or zone that is listed in the `Nodelist` property. For scalable resource groups, this node or zone is the first set of online nodes or zones that are listed in the `Nodelist` property, up to `Desired_primaries` or `Maximum_primaries`, whichever is less. If you specify the `-n` option and resource-group

affinities do exist, the affinities settings override the order of nodes in the `Nodelist` property. See the rg_properties(5) man page for more information about resource-group affinities.

Unlike the `switch` subcommand, this subcommand does not attempt to take any nodes or zones that are listed in the `Nodelist` property to the `Offline` state.

If you specify the `-e` option with this subcommand, all resources in the set of resource groups that are brought online are enabled.

You can specify the `-m` option to enable monitoring for all resources in the set of resource groups that are brought online. However, resources are not actually monitored unless they are first enabled and are associated with a `MONITOR_START` method.

You can also specify the `-M` option to indicate that all resource groups that are brought online are to be placed in a managed state. If the `-M` option is not specified, this subcommand has no effect on unmanaged resource groups.

Resource groups are brought online in an order that reflects resource and resource group dependencies. The order in which you specify resource groups on the command line does not matter.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`quiesce`

Brings the specified resource group to a quiescent state.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

This command stops a resource group from continuously switching from one node or zone to another node or zone if a `START` or `STOP` method fails.

Use the `-k` option to kill methods that are running on behalf of resources in the affected resource groups. If you do not specify the `-k` option, methods are allowed to continue running until they exit or exceed their configured timeout.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`remaster`

Switches the resource groups that you specify from their current primary nodes or zones to their most preferred nodes or zones. Preference order is determined by the `Nodelist` and `RG_affinities` properties.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

Unlike the `online` subcommand, this subcommand can switch resource groups offline from their current masters to bring them online on more preferred masters.

Resource groups are switched in an order that reflects resource group dependencies and affinities. The order in which you specify resource groups on the command line does not matter.

This subcommand has no effect on unmanaged resource groups.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`remove-node`
Removes a node or zone from the `Nodelist` property of a resource group.

You can use this subcommand only in the global zone.

After removing the node or zone, `remove-node` might reset the value of the `Maximum_primaries` or `Desired_primaries` property to the new number of nodes or zones in the `Nodelist` property. `remove-node` resets the value of the `Maximum_primaries` or `Desired_primaries` property only if either value exceeds the new number of nodes or zones in the `Nodelist` property.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`restart`
Takes a resource group offline and then back online on the same set of primary nodes or zones that currently host the resource group.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

If you specify the `-n` option, the resource group is restarted only on current masters that are in the list of nodes or zones that you specify.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`resume`
Resumes the automatic recovery actions on the specified resource group, which were previously suspended by the `suspend` subcommand.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

A suspended resource group is *not* automatically restarted or failed over until you explicitly issue the command that resumes automatic recovery. Whether online or offline, suspended data services remain in their current state. You can still manually switch the resource group to a different state on specified nodes or zones. You can also still enable or disable individual resources in the resource group.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`set`
Modifies the properties that are associated with the resource groups that you specify.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

You can modify the `Nodelist` property either with `-p Nodelist=-`*node*`--` or, as a convenience, with `-n` *node*.

You can also use the information in the *clconfigfile* file by specifying the `-i` option with the `set` subcommand. See the clconfiguration(5CL) man page.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`show`
Generates a configuration report, filtered by qualifier options, for resource groups that you specify.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resource groups that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resource groups.

You can use `-r` *resource* to include only those resource groups that contain resources. You can use `-t` *resourcetype* to include only those resource groups that contain a resource type in *resourcetype*. You can use `-n` *node* or `-n` *node:zone* to include only those resource groups that are online in one or more zones or on one or more nodes.

You can use the `-p` option to display a selected set of resource group properties rather than all resource group properties.

If you do not specify an operand or if you specify the plus sign operand (+), all resource groups, filtered by any qualifier options that you specify, are listed.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`status`
Generates a status report, filtered by qualifier options, for resource groups that you specify.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resource groups that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resource groups.

You can use `-r` *resource* to include only those resource groups that contain resources. You can use `-t` *resourcetype* to include only those resource groups that contain a resource type in *resourcetype*. You can use `-n` *node* or `-n` *node:zone* to include only those resource groups that are online in one or more zones or on one or more nodes.

If you specify `-s` *state*, only those groups with the states that you specify are listed.

Note – You can specify either the `-n` option or the `-s` option with the `status` subcommand. But, you cannot specify both options at the same time with the `status` subcommand.

If you do not specify an operand or if you specify the plus sign operand (+), all resource groups, filtered by any qualifier options that you specify, are listed.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`suspend`
Suspends the automatic recovery actions on and quiesces the specified resource group.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

A suspended resource group is *not* automatically restarted or failed over until you explicitly issue the command that resumes automatic recovery. Whether online or offline, suspended data services remain in their current state. You can still manually switch the resource group to a different state on specified nodes or zones. You can also still enable or disable individual resources in the resource group.

You might need to suspend the automatic recovery of a resource group to investigate and fix a problem in the cluster or perform maintenance on resource group services.

You can also specify the `-k` option to immediately kill methods that are running on behalf of resources in the affected resource groups. By using the `-k` option, you can speed the quiescing of the resource groups. If you do not specify the `-k` option, methods are allowed to continue running until they exit or they exceed their configured timeout.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`switch`
Changes the node or zone, or set of nodes or zones, that is mastering a resource group that you specify.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

Use the `-n` option to specify the list of nodes or zones on which to bring the resource groups online.

If a resource group is not already online, it is brought online on the set of nodes or zones that is specified by the `-n` option. However, groups that are online are brought offline on nodes or zones that are not specified by the `-n` option before the groups are brought online on new nodes or zones.

If you specify `-e` with this subcommand, all resources in the set of resource groups that are brought online are enabled.

You can specify `-m` to enable monitoring for all resources in the set of resource groups that are brought online. However, resources are not actually monitored unless they are first enabled and are associated with a `MONITOR_START` method.

You can specify the `-M` option to indicate that all resource groups that are brought online are to be placed in a managed state. If the `-M` option is not specified, this subcommand has no effect on unmanaged resource groups.

Resource groups are brought online in an order that reflects resource and resource group dependencies. The order in which you specify groups on the command line does not matter.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`unmanage`
Brings a resource group that you specify to an unmanaged state.

If you use this subcommand in a non-global zone, this subcommand successfully operates only on resource groups whose node list contains that zone. If you use this subcommand in the global zone, this subcommand can operate on any resource group.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

Options The following options are supported:

Note – Both the short and long form of each option is shown in this section.

`-?`
`--help`
Displays help information.

You can specify this option with or without a *subcommand*.

If you specify this option without a *subcommand*, the list of all available subcommands is displayed.

If you specify this option with a *subcommand*, the usage for that *subcommand* is displayed.

If you specify this option with the `create` or `set` subcommands, help information is displayed for all resource group properties.

If you specify this option with other options, with subcommands, or with operands, they are all ignored. No other processing occurs.

`-e`
`--enable`
Enables all resources within a resource group when the group is brought online.

You can use this option only with the `switch` and `online` subcommands.

`-F`
`--force`
Deletes a resource group and all of its resources forcefully, even if those resources are enabled or online. This option also removes both resources and resource groups from any dependency property settings or affinity property settings in other resources and in other resource groups.

Use the `-F` option with the `delete` subcommand with care. A forced deletion might cause changes to other resource groups that reference the deleted resource group, such as when a dependency or affinity is set. Dependent resources might be left with an invalid or error state after the forced deletion. If this occurs, you might need to reconfigure or restart the affected dependent resources.

`-i` {`-` | *clconfigfile*}
`--input={- |` *clconfigfile*`-}`
`--input {- |` *clconfigfile*`-}`
Specifies that you want to use the configuration information that is located in the *clconfigfile* file. See the clconfiguration(5CL) man page.

Specify a dash (`-`) with this option to provide configuration information through the standard input (`stdin`).

If you specify other options, they take precedence over the options and information in *clconfigfile*.

Only those resource groups that you specify are affected by this option.

`-k`
`--kill`
Kills RGM resource methods that are running on behalf of resources in the resource group that you specify.

You can use this option with the `quiesce` and `suspend` subcommands. If you do not specify the `-k` option, methods are allowed to continue running until they exit or they exceed their configured timeout.

`-m`
`--monitor`
Enables monitoring for all resources within a resource group when the resource group is brought online.

Resources, however, are not actually monitored unless they are first enabled and are associated with a `MONITOR_START` method.

You can use this option only with the `switch` and `online` subcommands.

`-M`
`--manage`
Specifies that all resource groups that are brought online by the `switch` or `online` subcommand are to be in a managed state.

`-n` *node*`-[:`*zone*`-[,…]]`
`--node=`*node*`-[:`*zone*`-[,…]]`
`--node` *node*`-[:`*zone*`-[,…]]`
Specifies a node, or zone, or a list of nodes or ones.

You can specify the name or identifier of a node for *node*. You can also specify a *zone* with *node*.

When used with the `list`, `show`, and `status` subcommands, this option limits the output. Only those resource groups that are currently online in one or more zones or on one or more nodes in the node list are included.

Specifying this option with the `create`, `add-node`, `remove-node`, and `set` subcommands is equivalent to setting the `Nodelist` property. The order of the nodes or zones in the `Nodelist` property specifies the order in which the group is to be brought online on those nodes or zones. If you do not specify a node list with the `create` subcommand, the `Nodelist` property is set to all nodes and zones in the cluster. The order is arbitrary.

When used with the `switch` and `online` subcommands, this option specifies the nodes or zones on which to bring the resource group online.

When used with the `evacuate` and `offline` subcommands, this option specifies the nodes or zones on which to bring the resource group offline.

When used with the `restart` subcommand, this option specifies nodes or zones on which to restart the resource group. The resource group is restarted on current masters which are in the specified list.

`-o` {`-` | *clconfigfile*}
`--output=`{`-` | *clconfigfile*`-`}

`--output {- | `*clconfigfile*`-}`
Writes resource group configuration information to a file or to the standard output (`stdout`). The format of the configuration information is described in the clconfiguration(5CL) man page.

If you specify a file name with this option, this option creates a new file. Configuration information is then placed in that file. If you specify `-` with this option, the configuration information is sent to the standard output (`stdout`). All other standard output for the command is suppressed.

You can use this option only with the `export` subcommand.

`-p` *name*
`--property=`*name*
`--property` *name*
Specifies a list of resource group properties.

You use this option with the `show` subcommand.

For information about the properties that you can set or modify with the `create` or `set` subcommand, see the description of the `-p` *name=value* option.

If you do not specify this option, the `show` subcommand lists most resource group properties. If you do not specify this option and you specify the `-verbose` option with the `show` subcommand, the subcommand lists all resource group properties.

Resource group properties that you can specify are described in "Resource Group Properties" in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS* in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*.

`-p` *name=value*
`-p` *name-+=array-values*
`-p` *name–=array-values*
`--property=`*name-=value*
`--property=`*name-+=array-values*
`--property=`*name-–=array-values*
`--property` *name-=value*
`--property` *name-+=array-values*
`--property` *name-–=array-values*
Sets or modifies the value of a resource group property.

You can use this option only with the `create` and `set` subcommands.

For information about the properties about which you can display information with the `show` subcommand, see the description of the `-p` *name* option.

Multiple instances of `-p` are allowed.

The operators to use with this option are as follows:

`=` Sets the property to the specified value. The `create` and `set` subcommands accept this operator.

`+=` Adds one or more values to a list of property values. Only the `set` subcommand accepts this operator. You can specify this operator only for properties that accept lists of string values, for example, `Nodelist`.

`–=` Removes one or more values to a list of property values. Only the `set` subcommand accepts this operator. You can specify this operator only for properties that accept lists of string values, for example, `Nodelist`.

`-r` *resource*[,…]
`--resource=`*resource*`-[,…]`
`--resource` *resource*`-[,…]`
Specifies a resource or a list of resources.

You can use this option only with the `list`, `show`, and `status` subcommands. This option limits the output from these commands. Only those resource groups that contain one or more of the resources in the resource list are output.

`-s` *state*[,…]
`--state=`*state*`-[,…]`
`--state` *state*`-[,…]`
Specifies a resource group state or a list of resource group states.

You can use this option only with the `list` and `status` subcommands. This option limits the output so that only those resource groups that are in the specified state on any specified nodes or zones are displayed. You can specify one or more of the following arguments (states) with this option:

`Error_stop_failed`
Any specified resource group that is in the `Error_stop_failed` state on any node or zone that you specify is displayed.

`Not_online`
Any specified resource group that is in any state other than `online` on any node or zone that you specify is displayed.

`Offline`
A specified resource group is displayed only if it is in the `Offline` state on *all* nodes and zones that you specify.

`Online`
Any specified resource group that is in the `Online` state on any node or zones that you specify is displayed.

`Online_faulted`
Any specified resource group that is in the `Online_faulted` state on any node or zone that you specify is displayed.

`Pending_offline`
Any specified resource group that is in the `Pending_offline` state on any node or zone that you specify is displayed.

`Pending_online`
Any specified resource group that is in the `Pending_online` state on any node or zone that you specify is displayed.

`Pending_online_blocked`
Any specified resource group that is in the `Pending_online_blocked` state on any node or zone that you specify is displayed.

`Unmanaged`
Any specified resource group that is in the `Unmanaged` state on any node or zone that you specify is displayed.

`-S`
`--scalable`
Creates a scalable resource group or updates the `Maximum_primaries` and `Desired_primaries` properties.

You can use this option only with the `create` and `add-node` subcommands.

When used with the `create` subcommand, this option creates a scalable resource group rather than a failover resource group. This option also sets both the `Maximum_primaries` and `Desired_primaries` properties to the number of nodes and zones in the resulting `Nodelist` property.

You can use this option with the `add-node` subcommand only if the resource group is already scalable. When used with the `add-node` subcommand, this option updates both the `Maximum_primaries` and `Desired_primaries` properties to the number of nodes and zones in the resulting `Nodelist` property.

You can also set the `RG_mode`, `Maximum_primaries`, and `Desired_primaries` properties with the `-p` option.

`-t` *resourcetype*[,…]
`--type=`*resourcetype*`-[,…]`
`--type` *resourcetype*`-[,…]`
Specifies a resource type or a list of resource types.

You can use this option only with the `list`, `show`, and `status` subcommands. This option limits the output from these commands. Only those resource groups that contain one or more of the resources of a type that is included in the resource type list are output.

You specify resource types as [*prefix*.]*type*[:*RT-version*]. For example, an `nfs` resource type might be represented as `SUNW.nfs:3.2`, `SUNW.nfs`, or `nfs`. You need to include an *RT-version* only if there is more than one version of a resource type that is registered in the cluster. If you do not include a *prefix*, `SUNW` is assumed.

`-T` *seconds*
`--time=`*seconds*
`--time` *seconds*
Specifies the number of seconds to keep resource groups from switching back onto a node or zone after you have evacuated resource groups from the node or zone.

You can use this option only with the `evacuate` subcommand. You must specify an integer value between 0 and 65535 for *seconds*. If you do not specify a value, 60 seconds is used by default.

Resource groups cannot fail over or automatically switch over onto the node or zone while that node or zone is being taken offline. This option also specifies that after a node or zone is evacuated, resource groups cannot fail over or automatically switch over for *seconds* seconds. You can, however, initiate a switchover onto the evacuated node or zone by using the `switch` and `online` subcommands before the timer expires. Only automatic switchovers are prevented.

`-v`
`--verbose`
Displays verbose information on the standard output (`stdout`).

`-V`
`--version`
Displays the version of the command.

If you specify this option with other options, with subcommands, or with operands, they are all ignored. Only the version of the command is displayed. No other processing occurs.

`-z` *zone*
`--zone=`*zone*
`--zone` *zone*
Applies the same zone name to all nodes in a node list for which a zone is not explicitly specified. You can specify this option only when you use the `-n` option.

Operands The following operands are supported:

resourcegroup	The name of the resource group that you want to manage.
+	All resource groups.

Exit Status The complete set of exit status codes for all commands in this command set are listed in the Intro(1CL) man page. Returned exit codes are also compatible with the return codes that are described in the scha_calls(3HA) man page.

If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`35` `CL_EIO`
I/O error

A physical input/output error has occurred.

`36` `CL_ENOENT`
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

`38` `CL_EBUSY`
Object busy

You attempted to remove a cable from the last cluster interconnect path to an active cluster node. Or, you attempted to remove a node from a cluster configuration from which you have not removed references.

`39` `CL_EEXIST`
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

Examples

EXAMPLE 1 Creating a New Failover Resource Group

The first command in the following example creates the failover resource groups `rg1` and `rg2`. The second command adds the resources that are included in the configuration file `cluster-1.xml` to these resource groups.

```
# clresourcegroup create rg1 rg2
# clresource create -g rg1,rg2 -i /net/server/export/cluster-1.xml +
```

EXAMPLE 2 Bringing All Resource Groups Online

The following command brings all resource groups online, with all resources enabled and monitored.

```
# clresourcegroup online -emM +
```

EXAMPLE 3 Adding a Node to the `Nodelist` Property

The following command adds the node `phys-schost-4` to the `Nodelist` property for all resource groups.

```
# clresourcegroup set -p Nodelist+=phys-schost-4 +
```

EXAMPLE 4 Adding a Zone to the `Nodelist` Property

The following command adds the zone `zone1` on node `phys-schost-4` to the `Nodelist` property for all resource groups.

```
# clresourcegroup set -p Nodelist+=phys-schost-4:zone1 +
```

EXAMPLE 5 Evacuating All Resource Groups From a Node

The following command evacuates all resource groups from the node `phys-schost-3`.

```
# clresourcegroup evacuate -n phys-schost-3 +
```

EXAMPLE 6 Evacuating All Resource Groups From a Zone

The following command evacuates all resource groups from the zone `zone1` on node `phys-schost-3`.

```
# clresourcegroup evacuate -n phys-schost-3:zone1 +
```

EXAMPLE 7 Bringing a Resource Group Offline on All Nodes and Zones

The following command brings the resource group `rg1` offline on all nodes and zones.

```
# clresourcegroup offline rg1
```

EXAMPLE 8 Refreshing an Entire Resource Group Manager Configuration

The first command in the following example deletes all resources and resource groups, even if they are enabled and online. The second command unregisters all resource types. The third command creates the resources that are included in the configuration file `cluster-1.xml`. The third command also registers the resources' resource types and creates all resource groups upon which the resource types depend.

```
# clresourcegroup delete --force +
# clresourcetype unregister +
# clresource -i /net/server/export/cluster-1.xml -d +
```

EXAMPLE 9 Listing All Resource Groups

The following command lists all resource groups.

```
# clresourcegroup list
rg1
rg2
```

EXAMPLE 10 Listing All Resource Groups With Their Resources

The following command lists all resource groups with their resources. Note that `rg3` has no resources.

```
# clresourcegroup list -v
Resource Group Resource
-------------- --------
rg1            rs-2
rg1            rs-3
rg1            rs-4
rg1            rs-5
rg2            rs-1
rg3            -
```

EXAMPLE 11 Listing All Resource Groups That Include Particular Resources

The following command lists all groups that include Sun Cluster HA for NFS resources.

```
# clresource list -t nfs
rg1
```

EXAMPLE 12 Clearing a `Start_failed` Resource State by Switching Over a Resource Group

The `Start_failed` resource state indicates that a `Start` or `Prenet_start` method failed or timed out on a resource, but its resource group came online anyway. The resource group

EXAMPLE 12 Clearing a `Start_failed` Resource State by Switching Over a Resource Group *(Continued)*

comes online even though the resource has been placed in a faulted state and might not be providing service. This state can occur if the resource's `Failover_mode` property is set to `None` or to another value that prevents the failover of the resource group.

Unlike the `Stop_failed` resource state, the `Start_failed` resource state does *not* prevent you or the Sun Cluster software from performing actions on the resource group. You do not need to issue the `reset` subcommand to clear a `Start_failed` resource state. You only need to execute a command that restarts the resource.

The following command clears a `Start_failed` resource state that has occurred on a resource in the `resource-grp-2` resource group. The command clears this condition by switching the resource group to the `schost-2` node.

```
# clresourcegroup switch -n schost-2 resource-grp-2
```

EXAMPLE 13 Clearing a `Start_failed` Resource State by Restarting a Resource Group

The following command clears a `Start_failed` resource state that has occurred on a resource in the `resource-grp-2` resource group. The command clears this condition by restarting the resource group on the `schost-1` node, which originally hosted the resource group.

```
# clresourcegroup restart resource-grp-2
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Interface Stability	Evolving

See Also

clresource(1CL), clresourcetype(1CL), cluster(1CL), Intro(1CL), `su`(1M), scha_calls(3HA), `attributes`(5), `rbac`(5), rg_properties(5), clconfiguration(5CL)

Notes

The superuser can run all forms of this command.

All users can run this command with the `-?` (help) or `-V` (version) option.

If you take a resource group offline with the `offline` subcommand, the `Offline` state of the resource group does not survive node reboots. In other words, if a node dies or joins the cluster, the resource group might come online on some node or zone, even if you previously switched the resource group offline. Even if all of the resources are disabled, the resource group will come online.

To prevent the resource group from coming online automatically, use the `suspend` subcommand to suspend the automatic recovery actions of the resource group. To resume automatic recovery actions, use the `resume` subcommand.

To run the `clresourcegroup` command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`add-node`	`solaris.cluster.modify`
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`evacuate`	`solaris.cluster.admin`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`manage`	`solaris.cluster.admin`
`offline`	`solaris.cluster.admin`
`online`	`solaris.cluster.admin`
`quiesce`	`solaris.cluster.admin`
`remaster`	`solaris.cluster.admin`
`remove-node`	`solaris.cluster.modify`
`restart`	`solaris.cluster.admin`
`resume`	`solaris.cluster.admin`
`set`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`
`status`	`solaris.cluster.read`
`suspend`	`solaris.cluster.admin`
`switch`	`solaris.cluster.admin`
`unmanage`	`solaris.cluster.admin`

Name clresourcetype, clrt – manage resource types for Sun Cluster data services

Synopsis

```
/usr/cluster/bin/clresourcetype   [subcommand -?]

/usr/cluster/bin/clresourcetype  -V

/usr/cluster/bin/clresourcetype subcommand
 [options] -v  [resourcetype]...

/usr/cluster/bin/clresourcetype add-node   [-z zone]
 -n node[:zone][,...] {+ |  resourcetype...}

/usr/cluster/bin/clresourcetype export  [-o  {-
 |  configfile}] [+ |  resourcetype...]

/usr/cluster/bin/clresourcetype list  [ [-z zone]
 -n node[:zone][,...]] [+ |  resourcetype...]

/usr/cluster/bin/clresourcetype list-props  [-p
 [name,]...] [+ |  resourcetype...]

/usr/cluster/bin/clresourcetype register  [-i  {-
 |  clconfiguration}] [ [-z zone]  { [-n node[:zone][,...]]
 |  -N}] [-f rtrfile] [-p  [name [+ |  -]=value,]...]
 {+ |  resourcetype...}

/usr/cluster/bin/clresourcetype remove-node  -nnode[:zone][,...]
 [-z zone] {+ |  resourcetype...}

/usr/cluster/bin/clresourcetype set  [ [-z zone]
 { [-n node[:zone][,...]] |  -N}] [-p  [name [+
 |  -]=value,]...] {+ |  resourcetype...}

/usr/cluster/bin/clresourcetype show  [ [-z zone]
 -n node[:zone][,...]] [+ |  resourcetype...]

/usr/cluster/bin/clresourcetype unregister  {+
 |  resourcetype...}
```

Description The `clresourcetype` command manages resource types for Sun Cluster data services. The `clrt` command is the short form of the `clresourcetype` command. The `clresourcetype` command and the `clrt` command are identical. You can use either form of the command.

For ease of administration, run this command from the global zone. You can run the subcommands `list`, `list-props`, `show`, and `export` from a non-global zone.

The general form of this command is as follows:

`clresourcetype` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the `-?` option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are provided with the description of the option in the OPTIONS section of this man page.

Subcommands The following subcommands are supported:

`add-node`
Adds the specified nodes to the node list for the resource types specified as operands to the command. Use this form of the command only in the global zone.

This subcommand accepts the plus sign (+) as an operand to specify all resource types.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand.

See also the description of the `remove-node` subcommand.

`export`
Exports the cluster resource-type configuration in the format that is described by the clconfiguration(5CL) man page. You can use this form of the command in a non-global zone.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list`
Displays a list of the resource types that are specified as operands to the command. By default, all resource types that are registered in the cluster are displayed. This subcommand accepts the plus sign (+) as an operand to specify all resource types that are registered in the cluster. You can use this form of the command in a non-global zone.

If you specify the `-n` *nodelist* option, only resource types that are registered for use on the nodes in *nodelist* are displayed.

If you specify the `-v` option, the node list of each resource type in the list is also displayed.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list-props`
Displays the resource type properties of the specified resource types. This subcommand accepts the plus sign (+) as an operand to specify all resource types that are registered in the cluster. You can use this form of the command in a non-global zone.

The `-p` option limits the set of properties that are to be displayed.

If you specify the `-v` option, the description of each property is also displayed.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`register`
Registers the resource types that are specified as operands to the command. A resource type must be registered before a resource of that type can be created. Use this form of the command only in the global zone.

The data service that defines each resource type must be installed on each node where the resource type is to be available. If the data service is installed on only a subset of cluster nodes, use the `-n` *nodelist* option to specify the subset of nodes. If the resource type is to be available on all nodes in the cluster, specify the `-N` option. When you use the `-N` option, the resource type is also available to any nodes that might be added to the cluster in the future. Omitting both the `-N` option and the `-n` *nodelist* option is equivalent to specifying the `-N` option. To specify the property name explicitly, use the `-p Installed_nodes=`*nodelist* option.

Information about a resource type that is registered with the cluster is obtained from the resource type registration (RTR) file that defines the resource type. The location and name of the RTR file typically follow these conventions:

- The RTR file is typically located in the `/opt/cluster/lib/rgm/rtreg` directory.
- The name of the RTR file typically matches the name of the resource type.

The location and file name of all RTR files that are supplied by Sun Microsystems, Inc. follow these conventions. For example, the RTR file that defines the `SUNW.nfs` resource type is contained in the file `/opt/cluster/lib/rgm/rtreg/SUNW.nfs`.

If an RTR file does not follow these conventions, you must specify the `-f` *rtrfile* option.

Caution – Do not register a resource type for which the `Global_zone` property is set to `TRUE` unless the resource type comes from a known and trusted source. Resource types for which this property is set to `TRUE` circumvent zone isolation and present a risk.

This subcommand accepts the plus sign (+) as an operand to specify all resource types that are not already registered. The complete list of available resource types is determined as follows:

- If you specify the `-i` *clconfiguration* option, *clconfiguration* defines the complete list of available resource types.
- If you do not specify the `-i` option, the complete list of available resource types contains only resource types that are supplied by Sun Microsystems, Inc. These resource types must also be installed on all nodes in the node list.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

See also the description of the `unregister` subcommand.

`remove-node`
Removes a node from the list of nodes for which the resource types in the operand list are registered. This subcommand accepts the plus sign (+) as an operand to specify all resource types that are registered in the cluster. Use this form of the command only in the global zone.

You can use this subcommand only on resource types that have already been registered for some nodes, but not all nodes, in a cluster. Consequently, an error occurs if you use this subcommand in the following situations:

- A resource type in the list of operands has already been registered for all nodes in a cluster. For information about the registration of resource types for all nodes in a cluster, see the description of the `-N` option.
- The `Installed_nodes` property of a resource type in the list of operands does not already specify a subset of the nodes in the cluster.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

See also the description of the `add-node` subcommand.

`set`

Sets properties of the resource types that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify all resource types that are registered in the cluster. Use this form of the command only in the global zone.

The only resource type property that you can set is the `Installed_nodes` property. You can modify this property by specifying the `-n` *nodelist* option without specifying the `-p` option. To specify the property name explicitly, use the `-p Installed_nodes=`*nodelist* option.

To limit the list of nodes on which the resource type is to be available, specify the `-n` *nodelist* option. Conversely, to specify that the resource type is to be available on all nodes in the cluster, specify the `-N` option. When you use the `-N` option, the resource type is also available to any nodes that might be added to the cluster in the future. You must specify the `-n` option or the `-N` option. If you omit both options, the subcommand does not change any configuration information.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`show`

Displays information about resource types that are registered in the cluster. By default, the following information for all resource types that are registered is displayed:

- The list of properties that is associated with each resource type
- Parameters that define these properties

If you specify the `-n` *nodelist* option, only resource types that are registered for use on nodes in *nodelist* are displayed.

If you specify the `-v` option, the following information is also displayed for each resource type:

- The methods that are defined for the resource type

- The timeout parameters of each method

You can use this form of the command in the non-global zone.

This subcommand accepts the plus sign (+) as an operand to specify all resource types that are registered in the cluster. If no operands are supplied, information about all resource types that are registered in the cluster is displayed.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`unregister`
Unregisters the resource types that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify all registered resource types for which no instances of the type exist. Use this form of the command only in the global zone.

Unregister a resource type before uninstalling the data service that defines the resource type.

If a resource of a certain resource type exists, you cannot unregister the resource type.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `register` subcommand.

Options The following options are supported:

`-?`
`--help`
Displays help information. When this option is used, no other processing is performed.

You can specify this option without a subcommand or with a subcommand.

If you specify this option without a subcommand, the list of subcommands for this command is displayed.

If you specify this option with a subcommand, the usage options for the subcommand are displayed.

`-f` *rtrfile*`-|`*rtrfiledir*
`--rtrfile=`*rtrfile*`-|`*rtrfiledir*
`--rtrfile` *rtrfile*`-|`*rtrfiledir*
Specifies the full path to an RTR file or a directory that contains RTR files for use in registering resource types. You can specify this option only with the `register` subcommand.

If you specify a file, you can register only one resource type.

You need to specify this option only if an RTR file that you are using does not follow these conventions:

- The RTR file is typically located in the `/opt/cluster/lib/rgm/rtreg` directory.
- The name of the RTR file typically matches the name of the resource type.

The location and file name of all RTR files that are supplied by Sun Microsystems, Inc follow these conventions. For example, the RTR file that defines the `SUNW.nfs` resource type is contained in the file `/opt/cluster/lib/rgm/rtreg/SUNW.nfs`.

If you use the `-i` option, you can specify a `resourcetypeRTRFile` element in the configuration information for any resource type that is specified in the configuration information. The `resourcetypeRTRFile` element specifies the RTR file that is to be used for registering the resource type. However, the `export` subcommand does not include the `resourcetypeRTRFile` element in generated configuration information. For more information about the `resourcetypeRTRFile` element, see the clconfiguration(5CL) man page.

`-i {-` | *clconfiguration*`-}`
`--input={-` | *clconfiguration*`-}`
`--input {-` | *clconfiguration*`-}`
Specifies configuration information that is to be used for registering resource types or for modifying the node lists of registered resource types. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through the standard input (`stdin`). To specify the standard input, specify `-` instead of a file name.

Only the resource types that are supplied as operands to the command are affected by this option. Options that are specified in the command override any options that are set in the `clconfiguration` file. If configuration parameters are missing in the `clconfiguration` file, you must specify these parameters at the command line.

`-N`
`--allnodes`
Specifies that the resource types in the list of operands are to be available on all nodes in the cluster. The `-N` option also makes these resource types available to any nodes that might be added to the cluster in the future. The option achieves this result by clearing the `Installed_nodes` property.

If you specify the `-N` option, you cannot specify the `-n` option in the same command.

You can specify the `-N` option only with the `register` subcommand or the `set` subcommand.

`-n` *node*`-[:`*zone*`-][,…]`
`--node=`*node*`-[:`*zone*`-][,...]`
`--node` *node*`-[:`*zone*`-][,...]`

Specifies a node or a list of nodes. You can specify each node as a node name or a node ID. For each node, you can optionally specify a non-global zone on the node.

If you specify the `-n` option, you cannot specify the `-N` option in the same command.

The subcommands with which you can specify this option are as follows:

`add-node`
Adds the specified nodes to the list of nodes for which resource types are registered.

`list`
Displays only resource types that are registered for use on the specified nodes.

`register`
Registers resource types only for use on the specified nodes. If you omit the `-n` option, the `register` subcommand registers resource types for use on all nodes. The subcommand also registers the resource types for any nodes that are added to the cluster in the future.

`remove-node`
Removes the specified nodes from the list of nodes for which resource types are registered.

`set`
Makes resource types available only on the specified nodes.

`show`
Displays information only about resource types that are registered for use on the specified nodes.

`-o {-` | *clconfiguration*`-}`
`--output={-` | *clconfiguration*`-}`
`--output {-` | *clconfiguration*`-}`
Specifies the location where configuration information about resource types is to be written. This location can be a file or the standard output (`stdout`). To specify the standard output, specify `-` instead of a file name. If you specify the standard output, all other standard output for the command is suppressed. You can specify this option only with the `export` subcommand.

Configuration information is written only for the resource types that are supplied as operands to the command. The information is written in the format that is defined in the clconfiguration(5CL) man page.

`-p` *name*`-=`*value*
`-p` *name*`-+=`*array-values*
`-p` *name*`--=`*array-values*
`--property=`*name*`-=`*value*
`--property` *name*`-=`*value*
`--property=`*name*`-+=`*array-values*

`--property` *name*`-+=`*array-values*
`--property=`*name*`--=`*array-values*
`--property` *name*`--=`*array-values*
Sets the value of a property for resource types that are supplied as operands to the command.

The operators to use with this option are as follows:

`=` Sets the property to the specified value.

`+=` Adds a value or values to a string array value. You can specify this operator only for properties that accept lists of string values, for example `Installed_nodes`.

`-=` Removes a value or values from a string array value. You can specify this operator only for properties that accept lists of string values, for example `Installed_nodes`

Using the option `-p Installed_nodes+=nodeC,nodeD` with the `set` subcommand is identical to using the option `-n nodeC,nodeD` with the `add-node` subcommand.

`-p` *name*`-[,…]`
`--property=`*name*`-[,...]`
`--property` *name*`-[,...]`
Specifies a list of properties for the `list-props` subcommand.

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The `-V` option displays only the version of the command. No other operations are performed.

`-v`
`--verbose`
Displays verbose messages to the standard output (`stdout`).

You can specify this option with any form of the command.

Do not specify the `-v` option with the `-o -` option. The `-v` option is ignored. The `-o -` option suppresses all other standard output.

`-z` *zone*
`--zone=`*zone*
`--zone` *zone*
Applies the same zone name to all nodes in a node list for which a zone is not explicitly specified. You can specify this option only with the `-n` option.

Operands Only the following operand is supported:

resourcetype Specifies the resource type that is to be managed or the resource types that are to be managed. If the subcommand accepts more than one resource, you can use the plus sign (+) to specify all resource types.

For a description of the format of resource type names, see "RGM Legal Names" in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS* in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

The following exit codes can be returned:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`18` `CL_EINTERNAL`
Internal error was encountered

An internal error indicates a software defect or other defect.

`35` `CL_EIO`
I/O error

A physical input/output error has occurred.

`36` `CL_ENOENT`
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

`41 CL_ETYPE`
Invalid type

The type that you specified with the `-t` or `-p` option does not exist.

These exit values are compatible with the return codes that are described in the scha_calls(3HA) man page.

Examples

EXAMPLE 1 Registering Resource Types

This example registers all resource types whose data services are installed on all nodes and that are not yet registered. The command runs in concise mode.

```
# clresourcetype register +
```

EXAMPLE 2 Registering Resource Types on Selected Nodes

This example registers all resource types whose data services are installed on node `phys-schost-1` and node `phys-schost-2` and that are not yet registered . The resources are to be made available only on these nodes. In this example, the command returns no error. The command runs in verbose mode.

```
# clresourcetype register -v -n phys-schost-1,phys-schost-2 +
```

EXAMPLE 3 Registering a Single Resource Type

This example registers the `SUNW.nfs:3.2` resource type. The data service for this resource type is installed on all cluster nodes.

```
# clresourcetype register nfs:3.2
```

EXAMPLE 4 Listing Resource Types

This example lists only the names of all registered resource types.

```
# clresourcetype list
SUNW.LogicalHostname
SUNW.SharedAddress
SUNW.nfs
SUNW.apache
```

EXAMPLE 5 Listing Resource Types With Their Node Lists

This example lists all registered resource types with their node lists.

```
# clresourcetype list -v

Resource Type         Node List
-------------         ---------
SUNW.LogicalHostname  <all>
SUNW.SharedAddress    <all>
SUNW.nfs                 phys-schost-1,phys-schost-2,phys-schost-3
SUNW.apache           phys-schost-1,phys-schost-2,phys-schost-3
```

EXAMPLE 6 Listing Resource Types on Specified Nodes

This example lists all resource types that are registered on `phys-schost-4`.

```
# clrt list -n phys-schost-4
SUNW.LogicalHostname
SUNW.SharedAddress
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), clreslogicalhostname(1CL), clresource(1CL), clresourcegroup(1CL), clressharedaddress(1CL), cluster(1CL), scha_calls(3HA), clconfiguration(5CL), r_properties(5), `attributes`(5), `rbac`(5)

Sun Cluster Data Services Planning and Administration Guide for Solaris OS

Notes

The superuser user can run all forms of this command.

Any user can run this command with the following options:

- `-?` option
- `-V` option

To run this command with subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
add-node	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
list-props	solaris.cluster.read
set	solaris.cluster.modify
register	solaris.cluster.modify
remove-node	solaris.cluster.modify
show	solaris.cluster.read
unregister	solaris.cluster.admin

Name clressharedaddress, clrssa – manage Sun Cluster resources for shared addresses

Synopsis

```
/usr/cluster/bin/clressharedaddress  [subcommand]
 -?

/usr/cluster/bin/clressharedaddress  -V

/usr/cluster/bin/clressharedaddress  [subcommand
 [options]] -v  [saresource]...

/usr/cluster/bin/clressharedaddress create  -g resourcegroup
 [-h lhost[,...] ] [-N netif@node[,...]]
 [-X node[:zone][,...]] [-p  name=value] [-d] saresource

/usr/cluster/bin/clressharedaddress create  -i {-
 |  clconfiguration} [-a] [-g resourcegroup[,...]]
 [-X node[:zone][,...]] [-p  name=value] [-d] {+
 |  saresource...}

/usr/cluster/bin/clressharedaddress delete  [-g resourcegroup[,...]]
 [-F] {+ |  saresource...}

/usr/cluster/bin/clressharedaddress disable  [-g resourcegroup[,...]]
 [-R] [ [-z zone]  -n node[:zone][,...]] {+ |  saresource...}

/usr/cluster/bin/clressharedaddress enable  [-g resourcegroup[,...]]
 [-R] [ [-z zone]  -n node[:zone][,...]] {+ |  saresource...}

/usr/cluster/bin/clressharedaddress export  [-o
 {- |  configfile}] [+ |  saresource...]

/usr/cluster/bin/clressharedaddress list  [-s state[,...]]
 [-g resourcegroup[,...]] [+ |  saresource...]

/usr/cluster/bin/clressharedaddress list-props
 [-l listtype] [-p name[,...]] [+ |  lhresource...]

/usr/cluster/bin/clressharedaddress monitor  [-g resourcegroup[,...]]
 {+ |  saresource...}

/usr/cluster/bin/clressharedaddress reset  [-f errorflag]
 [-g resourcegroup[,...]] {+ |  saresource...}

/usr/cluster/bin/clressharedaddress set  [-i  {-
 |  clconfiguration}] [-g resourcegroup[,...]]
 [-X node[:zone][,...]] [-p name[+|-]=value] {+
 |  saresource...}

/usr/cluster/bin/clressharedaddress show  [-g resourcegroup[,...]]
 [-p name[,...]] [+ |  saresource...]

/usr/cluster/bin/clressharedaddress status  [-s state[,...]]
 [ [-z zone]  -n node[:zone][,...]] [-g resourcegroup[,...]]
 [+ |  saresource...]
```

```
/usr/cluster/bin/clressharedaddress unmonitor  [-g resourcegroup[,...]]
 {+ |  saresource...}
```

Description The `clressharedaddress` command manages resources for Sun Cluster shared addresses. The `clrssa` command is the short form of the `clressharedaddress` command. The `clressharedaddress` command and the `clrssa` command are identical. You can use either form of the command.

You can also use the clresource(1CL) command to manage resources for a shared address.

You can run the `clressharedaddress` command with the `create` subccommand or the `delete` subcommand only from the global zone.

Some subcommands of the `clressharedaddress` command modify the resource configuration. You can use these subcommands from the global zone or a non-global zone. If you use a subcommand that modifies resource configuration from a non-global zone, only resources that the non-global zone can master are modified. The following subcommands modify resource configuration:

- `disable`
- `enable`
- `monitor`
- `reset`
- `set`
- `unmonitor`

Some subcommands of the `clressharedaddress` command only obtain information about resources. You can use these subcommands from the global zone or a non-global zone. However, even when you use these subcommands from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources. The following commands only obtain information about resources:

- `export`
- `list`
- `list-props`
- `show`
- `status`

To avoid unpredictable results from this command, run all forms of the command from the global zone.

The general form of this command is:

`clressharedaddress` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the option `-?` or `-V`.

Each option of this command has a long form and a short form. Both forms of each option are given with the description of the option in the OPTIONS section of this man page.

Subcommands The following subcommands are supported:

`create`
Creates the shared address resources that are specified as operands to the command.

When you use`create` with the `-i` option to specify a configuration file, the subcommand accepts the plus sign (+) as an operand. When you use the + operand, all resources in the configuration file that do not exist are created.

By default, resources are created in the enabled state with monitoring enabled. However, a resource comes online and is monitored only after the resource's resource group is brought online. To create resources in the disabled state, specify the `-d` option.

You can use this subcommand only in the global zone.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand.

See also the description of the `delete` subcommand.

`delete`
Deletes the shared address resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are deleted.

The `-g` option filters the list of operands to limit the resources that are deleted. The `-g` option deletes only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

- By default, a resource is deleted *only* if the following conditions are met:
- The resource must be disabled.
- All dependencies on the resource must be eliminated.
- To ensure that all specified resources are deleted, specify the `-F` option. The effects of the `-F` option are as follows:
- All specified resources are deleted, even resources that are not disabled.
- All specified resources are removed from resource-dependency settings of other resources.

Resources are deleted in the order that is required to satisfy dependencies between the resources, regardless of the order in which resources are specified on the command line.

You can use this subcommand only in the global zone.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

See also the description of the `create` subcommand.

`disable`
Disables the shared address resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are disabled.

The `-g` option filters the list of operands to limit the resources that are disabled. The `-g` option disables only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

To ensure that all required resource dependencies are satisfied, specify the `-R` option. The `-R` option disables any resources that depend on the resources that are specified as operands to the command, even if the resources are not specified as operands to the command. The `-g` option and the `-t` option do not apply to resources that are to be disabled solely to satisfy resource dependencies.

Resources are disabled in the order that is required to satisfy dependencies between the resources, regardless of the order in which resources are specified on the command line.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `enable` subcommand.

`enable`
Enables the shared address resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that all resources are enabled.

The `-g` option filters the list of operands to limit the resources that are enabled. The `-g` option enables only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

To ensure that all required resource dependencies are satisfied, specify the `-R` option. The `-R` option enables any resources that depend on the resources that are specified as operands to the command, even if the resources are not specified as operands to the command. The `-g` option does not apply to resources that are to be enabled solely to satisfy resource dependencies.

Resources are enabled in the order that is required to satisfy dependencies between the resources, regardless of the order in which resources are specified on the command line.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `disable` subcommand.

`export`
Exports the shared address resource configuration in the format that is described by the clconfiguration(5CL) man page.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list`
Displays a list of the shared address resources that are specified as operands to the command. By default, all resources are displayed.

The `-g` option filters the list of operands to limit the resources that are displayed. The `-g` option displays only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

This subcommand accepts the plus sign (+) as an operand to specify all resources in the specified resource groups or that are instances of the specified resource types. If no operands are supplied, all resources in the specified resource groups or that are instances of the specified resource types are displayed.

If you specify the `-v` option, the resource group and resource type of each resource in the list is also displayed.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`list-props`
Displays a list of the properties of the shared address resources that are specified as operands to the command. By default, the extension properties of all resources are displayed.

The following options filter the list of operands to limit the resources whose properties are displayed:

`-g` *resourcegrouplist*	Displays the properties only of the shared address resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

The `-l` option specifies the type of resource properties that are to be displayed:

`-l all`	Specifies that standard properties and extension properties are displayed.
`-l extension`	Specifies that only extension properties are displayed. By default, only extension properties are displayed.
`-l standard`	Specifies that only standard properties are displayed.

If you do not specify the `-l`option, only extension properties are displayed, unless you specify a standard property explicitly by using the `-p` option or the `-y` option.

The `-p` option limits the set of resource properties that is to be displayed. The `-p` option displays only the properties that are specified in *namelist*. You can specify standard properties and extension properties in *namelist*.

If you specify the `-v` option, the description of each property is also displayed.

This subcommand accepts the plus sign (+) as an operand to specify all resources in the specified resource groups or that are instances of the specified resource types. If no operands are supplied, properties of all resources in the specified resource groups or that are instances of the specified resource types are displayed.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`monitor`

Turns on monitoring for the shared address resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that monitoring is turned on for all resources.

The `-g` option filters the list of operands to limit the resources that are monitored. The `-g` option monitors only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

If monitoring is turned on for a resource, the resource is monitored only if the following conditions are met:

- The resource is enabled.
- The resource group that contains the resource is online on at minimum one cluster node.

Note – When you turn on monitoring for a resource, you do *not* enable the resource.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `unmonitor` subcommand.

`reset`
Clears an error flag that is associated with the shared address resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that the error flag is cleared for all resources.

The `-g` option filters the list of operands to limit the resources that are reset. The `-g` option resets only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

By default, the `reset` subcommand clears the `STOP_FAILED` error flag. To specify explicitly the error flag that is to be cleared, use the `-f` option. The only error flag that the `-f` option accepts is the `STOP_FAILED` error flag.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

`set`
Modifies specified properties of the shared address resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that the specified properties of all resources are modified.

The `-g` option filters the list of operands to limit the resources that are modified. The `-g` option modifies only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`show`
Displays the configuration of the shared address resources that are specified as operands to the command. By default, the configuration of all resources is displayed.

The `-g` option filters the list of operands to limit the resources for which the configuration is displayed. The `-g` option displays the configuration of only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

The `-p` option limits the set of resource properties that is to be displayed. The `-p` option displays only the properties that are specified in *namelist*. You can specify standard properties and extension properties in *namelist*.

This subcommand accepts the plus sign (+) as an operand to specify all resources in the specified resource groups or that are instances of the specified resource types. If no operands are supplied, the configuration of all resources in the specified resource groups or that are instances of the specified resource types are displayed.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`status`
Displays the status of the shared address resources that are specified as operands to the command. By default, the status of all resources is displayed.

The following options filter the list of operands to limit the list of resources for which the status is displayed:

`-g` *resourcegrouplist*	Displays the status of only the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.
`-n` *nodelist*	Displays the status of only the resources in the list of operands that are hosted on the nodes or in the zones in *nodelist*.
`-s` *statelist*	Displays the status of only the resources in the list of operands that are in the states in *statelist*.

This subcommand accepts the plus sign (+) as an operand to specify all resources in the specified resource groups or that are instances of the specified resource types. If no

operands are supplied, the status of all resources in the specified resource groups or that are instances of the specified resource types are displayed.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, the scope of the command is *not* limited to that zone. Information about all resources that are supplied as operands to the command is obtained, regardless of whether the non-global zone can master the resources.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand.

`unmonitor`
Turns off monitoring for the shared address resources that are specified as operands to the command. This subcommand accepts the plus sign (+) as an operand to specify that monitoring is turned off for all resources.

If you turn off monitoring for a resource that is disabled, the resource is not affected. The resource and its monitor are already offline.

Note – When you turn off monitoring for a resource, you do *not* disable the resource. However, when you disable a resource, you do not need to turn off monitoring for the resource. The disabled resource and its monitor are kept offline.

The `-g` option filters the list of operands to limit the resources for which monitoring is turned off. The `-g` option turns off monitoring for the resources in the list of operands that are members of the resource groups in *resourcegrouplist*.

You can use this subcommand from the global zone or a non-global zone. If you use this subcommand from a non-global zone, only resources that the non-global zone can master are modified.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand.

See also the description of the `disable` subcommand and the `monitor` subcommand.

Options The following options are supported:

`-?`
`--help`
Displays help information. When this option is used, no other processing is performed.

You can specify this option without a subcommand or with a subcommand.

If you specify this option without a subcommand, the list of subcommands for this command is displayed.

If you specify this option with a subcommand, the usage options for the subcommand are displayed.

The effect of this option with specific subcommands is as follows:

`create`	When specified with the `-g` option, this option displays help information for all resource properties of the specified resource group.
`set`	Displays help information for properties of the resources that are specified as operands to the command.

`-a`
`--automatic`

Automatically performs the following additional operations when resources are being created from cluster configuration information:

- Registering resource types
- Creating resource groups
- Creating resources on which the resources that are specified in the list of operands depend

The cluster configuration information must contain sufficient information to do all of the following:

- Enable the resource types to be registered
- Enable the resource groups to be created
- Enable the resources to be create

You can specify this option only with the `create` subcommand. If you specify this option, you must also specify the `-i` option and provide a configuration file.

`-d`
`--disable`

Disables a resource when the resource is created. You can specify this option only with the `create` subcommand. By default, resources are created in the enabled state.

Enabling a resource does not guarantee that the resource is brought online. A resource comes online only after the resource's resource group is brought online on at minimum one node.

`-f` *errorflag*
`--flag` *errorflag*

Specifies explicitly the error flag that is to be cleared by the `reset` subcommand. You can specify this option only with the `reset` subcommand. By default, the `reset` subcommand clears the `STOP_FAILED` error flag.

The only error flag that the `-f` option accepts is the `STOP_FAILED` error flag.

`-F`
`--force`

Forces the deletion of resources that are not disabled. You can specify this option only with the `delete` subcommand.

`-g` *resourcegroup*[,…]
`--resourcegroup` *resourcegroup*`-[,…]`
Specifies a resource group or a list of resource groups.

For subcommands except `create`, the command acts on only the resources in the list of operands that are members of the resource groups that the `-g` option specifies.

The effect of this option with specific subcommands is as follows:

`create`	Specifies that the resource is created in the specified resource group. When you use `-g` with the `create` subcommand, you can specify only one resource group.

`-h` *lhost*[,…]
`--logicalhost` *lhost*`-[,…]`
Specifies the host name list. You must use the `-h` option either when more than one logical host needs to be associated with the new `SharedAddress` resource or when the logical host does not have the same name as the resource itself. All logical hosts in a `HostnameList` for a `SharedAddress` resource must be on the same subnet. If you do not specify the `HostnameList` property, the `HostnameList` will be the same as the `SharedAddress` resource.

The logical host names for a `SharedAddress` resource must be on the same subnet.

You can use `-h` can instead of setting the `HostnameList` property with `-p`; however, you cannot use `-h` and explicitly set `HostnameList` in the same command.

You can only use `-h` is with the `create` subcommand.

`-i` {`-` | *clconfiguration*}
`--input {- |` *clconfiguration*`-}`
Specifies configuration information that is to be used for creating or modifying shared address resources. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through standard input. To specify standard input, specify `-` instead of a file name.

Only the resources that are supplied as operands to the command are created or modified. Options that are specified in the command override any options that are set in the configuration information. If configuration parameters are missing in the configuration information, you must specify these parameters on the command line.

The effect of this option with specific subcommands is as follows:

`create`	When specified with the `-a` option, this option registers all required resource types and creates all required resource groups. You must supply all information that is

required for the registration and configuration. All other configuration data is ignored.

`-l` *listtype*
`--listtype` *listtype*
Specifies the type of resource properties that are to be displayed by the `list-props` subcommand. You can specify this option only with the `list-props` subcommand.

You must specify one value from the following list for *listtype*:

`all`	Specifies that standard properties and extension properties are displayed.
`extension`	Specifies that only extension properties are displayed. By default, only extension properties are displayed.
`standard`	Specifies that only standard properties are displayed.

If you do not specify the `-l` option, only extension properties are displayed, unless you specify a standard property explicitly by using the `-p` option.

`-n` *node*[:*zone*][,…]
`--node` *node*`-[:`*zone*`-][,…]`
Specifies a node or a list of nodes. You can specify each node as node name or a node ID. For each node, you can optionally specify a non-global zone on the node.

The subcommands with which you can specify this option are as follows:

`disable`	Disables only the resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`enable`	Enables only the resources in the list of operands that are hosted on the specified nodes or in the specified zones.
`status`	Reports the status only of resources in the list of operands that are hosted on the specified nodes or in the specified zones.

`-N` *netif@node*[,…]
`--netiflist` *netif*`-@`*node*`-[,…]`
Specifies a resource property. The `-N`option enables you to set the `NetIfList` property without using the `-p` option for the property. If you do not specify `-N`, the `clressharedaddress` command attempts to set the `NetIfList` property for you based on available IPMP groups or public adapters, and the subnet associated with the `HostnameList` property.

You can specify the `NetIfList` property in the form of *ipmpgroup@node*[,…]. However, `-N` accepts both *ipmpgroup@node*[,…] and *publicNIC@node*[,…]. If you do not use `-N`, or if you use it with *publicNIC@node*, the `clressharedaddress` command attempts to create

the necessary IPMP groups. The system creates a set of one or more single-adapter IPMP groups with a set of default later modified to include multiple adapters using standard Solaris interfaces.

You can use `-N` instead of directly setting the `NetIfList` property with `-p`; however, you cannot use `-N` and explicitly set `NetIfList` in the same command.

You can only use `-N` with the `create` subcommand.

`-o` {`-` | *clconfiguration*}
`--output {- |` *clconfiguration*`-}`
Specifies the location where resource configuration information is to be written. This location can be a file or standard output. To specify standard output, specify `-` instead of a file name. If you specify standard output, all other standard output for the command is suppressed. You can specify this option only with the `export` subcommand.

Configuration information is written only for the resources that are supplied as operands to the command. The information is written in the format that is defined in the clconfiguration(5CL) man page.

`-p` *name=value*
`-p` *name-+=array-values*
`-p` *name-=array-values*
`--property` *name-=value*
`--property` *name+=array-values*
`--property` *name--=array-values*
Sets the standard properties and extension properties of a resource. You can specify this option only with the `create` subcommand and the `set` subcommand.

For a description of standard properties, see the r_properties(5) man page.

For a description of a resource type's extension properties, see the documentation for the resource type.

The operators to use with this option are as follows:

`=` Sets the property to the specified value. The `create` subcommand and the `set` subcommand accept this operator.

`+=` Adds a value or values to a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for string array values.

`-=` Removes a value or values from a string array value. Only the `set` subcommand accepts this operator. You can specify this operator only for string array values.

If a per-node property is to be set only on a subset of cluster nodes, specify the nodes where the property is set by appending the list of nodes in braces to the property name as follows:

name{*nodelist*}

nodelist is a comma-separated list of node names or node IDs. For more information about per-node properties, see the rt_properties(5) man page.

`-p` *name*[,…]
`--property` *name*`-[,…]`
Specifies a list of properties for the `list-props` subcommand and `show` subcommand.

You can use this option for standard properties and extension properties of a resource.

For a description of standard properties, see the r_properties(5) man page.

For a description of a resource type's extension properties, see the documentation for the resource type.

Without this option, the `list-props` subcommand and `show` subcommand list all or most resource properties, depending on whether the `-v` option is also specified.

`-R`
`--recursive`
Recursively enables or disables resources to ensure that all required dependencies are satisfied. You can specify this option only with the `disable` subcommand and the `enable` subcommand.

The effect of this option with these subcommands is as follows:

`disable`	Disables any resources that depend on the resources that are specified as operands to the command, even if the resources are not specified as operands to the command.
`enable`	Enables any resources on which resources that are specified as operands to the command depend, even if the resources are not specified as operands to the command.

`-s` *state*[,…]
`--state` *state*`-[,…]`
Specifies a list of states for the `list` subcommand and `status` subcommand.

This option limits the output to include only those resources that are in one of the specified states on one or more nodes in the node list.

The possible states are as follows:

- `degraded`
- `detached`
- `faulted`
- `monitor_failed`
- `not_online` - specifies any state other than `online` or `online_not_monitored`
- `offline`
- `online`

- online_not_monitored
- start_failed
- stop_failed
- unknown
- unmonitored
- wait

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The `-V` option only displays the version of the command. No other operations are performed.

`-v`
`--verbose`
Displays verbose messages to standard output.

You can specify this option with any form of the command.

Do not specify the `-v` option with the `-o -` option. The `-v` option is ignored. The `-o -` option suppresses all other standard output.

`-X` *node*[:*zone*][,…]
`--auxnode` *node*`-[:`*zone*`-][,…]`
Sets the `AuxNodeList` `SharedAddress` resource property.

The nodes in the `AuxNodeList` list can host the set of logical hosts that is associated with the shared address resource. However, these nodes cannot serve as the primary node during a failover.

`-z` *zone*
`--zone` *zone*
Applies the same zone name to all nodes in a node list for which a zone is not explicitly specified. You can specify this option only with the `-n` option.

Operands The following operands are supported:

resource Specifies that the Sun Cluster resource names should be accepted as operands. If the subcommand accepts more than one resource, you can use the plus sign (+) to specify all shared address resources.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`9` `CL_ESTATE`
Object is in wrong state

You tried to modify a property, a resource group, or other object that you cannot modify at that particular time or at any time.

`10` `CL_EMETHOD`
Resource method failed

A method of a resource failed. The method failed for one of the following reasons:

- The `validate` method failed when you tried to create a resource or modify the properties of a resource.
- A method other than `validate` failed when you tried to enable, disable, or delete a resource.

`15` `CL_EPROP`
Invalid property

The property or value that you specified with the `-p`, `-y`, or `-x` option does not exist or is not allowed.

`35` `CL_EIO`
I/O error

A physical input/output error has occurred.

`36` `CL_ENOENT`
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

39 `CL_EEXIST`
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

These exit values are compatible with the return codes that are described in the scha_calls(3HA) man page.

Examples

EXAMPLE 1 Creating a Shared Address Resource

This command creates a resource that is named `sharedhost1` in a resource group that is named `rg-failover`. The resource is created in the enabled state, with monitoring enabled.

```
# clressharedaddress create -g rg-failover sharedhost1
```

EXAMPLE 2 Creating a Shared Address Resource With a Different Logical Host Name

This command creates a resource named `rs-sharedhost1` in a resource group that is named `rg-failover`.

The logical host name is not the same as the resource name, but the name and IP address of the logical host remain the same.

```
# clressharedaddress create -g rg-failover \
-h sharedhost1 rs-sharedhost1
```

EXAMPLE 3 Specifying the IPMP Groups for a Shared Address Resource

This command sets the IPMP groups for the `sharedhost1` resource.

```
# clressharedaddress create -g rg-failover \
-N ipmp0@black,ipmp0@white sharedhost1
```

EXAMPLE 4 Deleting a Shared Address Resource

This command deletes a resource that is named `sharedhost1`.

```
# clressharedaddress delete sharedhost1
```

EXAMPLE 5 Listing Shared Address Resources

This command lists all shared address resources.

```
# clressharedaddress list
sharedhost1
sharedhost2
```

EXAMPLE 6 Listing Shared Address Resources With Resource Groups and Resource Types

This command lists all shared address resources with their resource groups and resource types.

```
# clressharedaddress list -v
Resources   Resource Groups Resource Types
---------   --------------- --------------
sharedhost1 rg-failover-1   SUNW.SharedAddress
sharedhost2 rg-failover-2   SUNW.SharedAddress
```

EXAMPLE 7 Listing Extension Properties of Shared Address Resources

This command lists the extension properties of all shared address resources.

```
# clressharedaddress list-props -v
Properties       Descriptions
----------       ------------
NetIfList        List of IPMP groups on each node
AuxNodeList      List of nodes on which this resource is available
HostnameList     List of hostnames this resource manages
CheckNameService Name service check flag
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cluster(1CL), clresource(1CL), clreslogicalhostname(1CL), clresourcegroup(1CL), clresourcetype(1CL), scha_calls(3HA), clconfiguration(5CL), `rbac`(5), r_properties(5)

Notes

The superuser user can run all forms of this command.

Any user can run this command with the following options:

- -? option

- -V option

To run this command with subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
create	solaris.cluster.modify
delete	solaris.cluster.modify
disable	solaris.cluster.admin
enable	solaris.cluster.admin
export	solaris.cluster.read
list	solaris.cluster.read
list-props	solaris.cluster.read
monitor	solaris.cluster.admin
reset	solaris.cluster.admin
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

Name clsetup – configure Sun Cluster interactively

Synopsis

```
/usr/cluster/bin/clsetup  -V

/usr/cluster/bin/clsetup  -?

/usr/cluster/bin/clsetup  [-f logfile]
```

Description The `clsetup` command provides the following configuration capabilities, depending on what state the cluster is in when you issue the command. The user must be `superuser` to run this command.

This command has no short form.

- When you run the `clsetup` command at post-installation time, the command performs initial setup tasks, such as configuring quorum devices and resetting the `installmode` property. If you deselected automatic Quroum Configuration when you created the cluster using either `scinstall` or the `cluster create` command, then you must run the `clsetup` command immediately after the cluster has been installed. Ensure that all nodes have joined the cluster before you run the `clsetup` command and reset the `installmode` property.

  If you used automatic quorum configuration when you created the cluster, you do not need to run the `clsetup` command after cluster installation. The automatic quorum configuration feature also resets the `installmode` property of the cluster.

  You can issue this form of the `clsetup` command from any node in the cluster.
- When you run the `clsetup` command during normal cluster operation, the `scsetup` command provides an interactive, menu-driven utility to perform cluster configuration tasks. The following are some of the cluster components which this utility administers:
  - Quorum
  - Resource groups
  - Data services
  - Cluster interconnect
  - Device groups and volumes
  - Private hostnames
  - New nodes
  - Other cluster properties

  You can issue this form of the `clsetup` command from any node in the cluster.
- When you run the `clsetup` command from a node that is in noncluster mode, the `clsetup` command provides a menu-driven utility for changing and displaying the private IP address range.

  You must reboot all nodes into noncluster mode before you start this form of the `clsetup` utility.

Options The following options are supported:

`-?`
`--help`
Prints help information for the command.

`-f` *logfile*
`--file` *logfile*
Specifies the name of a log file to which commands can be logged. If this option is specified, most command sets generated by `clsetup` can br run and logged, or just logged, depending on user responses.

`-V`
`--version`
Prints the version of the command set. No command line processing will be performed and the command will not enter into its interactive menu.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also Intro(1CL), cldevicegroup(1CL), clnode(1CL), clquorum(1CL), clreslogicalhostname(1CL), clresourcegroup(1CL), clresourcetype(1CL), clressharedaddress(1CL), cluster(1CL), cltelemetryattribute(1CL)

Name clsnmphost – administer list of Sun Cluster SNMP hosts

Synopsis

```
/usr/cluster/bin/clsnmphost  -V

/usr/cluster/bin/clsnmphost  [subcommand] -?

/usr/cluster/bin/clsnmphost  [subcommand  [options]]
 -v [host]

/usr/cluster/bin/clsnmphost add  [-c  community[,...]]
 [-n node,...] host [...]

/usr/cluster/bin/clsnmphost add  -i {- |  clconfigfile}
 [-c  community[,...]] [-n node[,...]] host [...]

/usr/cluster/bin/clsnmphost export  [-o {- |  clconfigfile}]
 [-c community[,...]] [-n node[,...]] [+
 |  host...]

/usr/cluster/bin/clsnmphost list  [-c community[,...]]
 [-n  node[,...]] [+ |  host...]

/usr/cluster/bin/clsnmphost remove  [-c community[,...]]
 [-n node[,...]] {+ |  host...}

/usr/cluster/bin/clsnmphost show  [-c community[,...]]
 [-n node[,...]] [+ |  host...]
```

Description The `clsnmphost` command administers Simple Network Management Protocol (SNMP) hosts and community names that will receive notifications of SNMP events. The SNMP hosts use the cluster Management Information Bases (MIBs) to provide access control mechanisms. When a MIB sends SNMP trap notifications, the SNMP hosts that were configured with this command can identify the hosts to which to send trap notifications. For more information about the cluster MIBs, see the clsnmpmib(1CL) man page.

This command has no short form.

The general form of this command is as follows:

`clsnmphost` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the option `-?` or `-V`.

Each option of this command has a long form and a short form. Both forms of each option are provided with the description of the option in the OPTIONS section of this man page.

See the Intro(1CL) man page for more information.

You can use this command only in the global zone.

Subcommands The following subcommands are supported:

`add`
Adds an SNMP host to the list of hosts that will receive trap notifications for the cluster MIBs and will be able to access the MIB tables.

You can use this subcommand only in the global zone.

If you use the `add` subcommand without the `-n` option, only the current node is affected. If you use `add` without the `-c` option, the subcommand uses `public` as the default community name. Specify the host by using either an IP address or host name.

If the specified community name does not exist, the command creates the community. Use the `-i` option to import one or more host configurations from the *clconfigfile*.

Users other than superuser require `solaris.cluster.modify` role–based access control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

`export`
Exports the SNMP host information for the specified node.

You can use this subcommand only in the global zone.

Use the `-n` option to specify one or more nodes for which to export the SNMP host information. If you use `export` without the `-n` option, the subcommand exports only SNMP host information for the current node.

For more information about the output format from the `export` subcommand, see the clconfiguration(5CL) man page. By default, all output is sent to standard output. Use the `-o` option followed by a file name to redirect the output to the file.

By using the `-c` option, you can limit the output from the `export` subcommand to information only for hosts in a particular community. Specify one or more hosts as operands to restrict the output information to only these hosts.

Users other than superuser require`solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`list`
Lists the SNMP hosts that are configured on the specified node.

You can use this subcommand only in the global zone.

If you use the `list` subcommand without the `-n` option, only SNMP hosts on the current node are listed. By default this subcommand lists all hosts on the node. To restrict the output to information about specific hosts, specify one or more hosts as operands. You can also use the `-c` option to list only those hosts in the specified community.

Users other than `superuser` require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`remove`
Removes an SNMP host from the node configuration.

You can use this subcommand only in the global zone.

To remove a host from the configuration, you must specify the host name as an operand. If you use the `remove` subcommand without the `-n` option, only SNMP hosts on the current node are removed. To remove all hosts, use the plus sign (+) sign. To remove one or more hosts from a specific community, use the `-c` option.

Users other than `superuser` require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`show`
Shows the SNMP host information on the specified node.

You can use this subcommand only in the global zone.

If you use the `show` subcommand without the `-n` option, only information for SNMP hosts on the current node is displayed. By default, the `show` subcommand displays information for all the hosts and their communities. To restrict the output to information about only specific hosts in a community, use the `-c` option, or specify the name of one or more hosts as operands.

Users other than `superuser` require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

Options The following options are supported:

`-?`
`--help`
Prints help information.

You can specify this option with or without a subcommand.

- If you use this option without a subcommand, the list of available subcommands is displayed.
- If you use this option with a subcommand, the usage options for that subcommand are displayed.

When this option is used, no other processing is performed.

`-c` *community*
`--community` *community*
Specifies the SNMP community name that will be used in conjunction with a host name. This option might also be used with other subcommands to narrow the scope of the subcommand operation. For example, when used with the `remove` subcommand, the `-c` option can be used to remove one or many hosts from a specific *community*. If you use the `add` subcommand without the `-c` option, the subcommand uses `public` as the default community name.

`-i` {`-` | *clconfigfile*}
`--input {- |` *clconfigfile*`-}`
Specifies configuration information that can be used for validating or modifying the SNMP hosts configuration. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through standard input. To specify standard input, specify the minus sign (`-`) instead of a file name.

`-n` *node*[,…]
`--node[s]` *node*`-[,…]`
Specifies a node or a list of nodes. You can specify each node as node name or as a node ID. All forms of the `clsnmphost` command accept this option.

`-o` {`-` | *clconfigfile*}
`--output {- |` *clconfigfile*`-}`
Writes cluster SNMP host configuration information in the format that is defined by the clconfiguration(5CL) man page. This information can be written to a file or to standard output.

To write to standard output, specify the minus sign (`-`) instead of a file name. If you specify standard output, all other standard output for the command is suppressed.

If you supply a file name, the configuration is copied to a new file of that name.

The `-o` option is valid only with the `export` subcommand. If you do not specify the `-o` option , the output will print to standard output.

`-V`
`--version`
Prints the version of the command.

Do not specify this option with subcommands, operands, or other options because they are ignored. The `-V` option displays only the version of the command. No other operations are performed.

`-v`
`--verbose`
Prints verbose information to standard output.

You can specify this option with any form of the command, although some subcommands might not produce expanded output. For example, the `export` subcommand does not produce expanded output when you specify the verbose option.

Operands The following operands are supported:

+	Specifies all SNMP host entries.
host	Specifies the IP address, IPv6 address, or name of a host that is provided access to the SNMP MIBs on the cluster.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`18` `CL_EINTERNAL`
Internal error was encountered

An internal error indicates a software defect or other defect.

`35` `CL_EIO`
I/O error

A physical input/output error has occurred.

`36` `CL_ENOENT`
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

Examples

EXAMPLE 1 Adding a Host by Specifying the Host Name

The following command adds the host `myhost` to the SNMP host list of the community on the current node `private`.

```
# clsnmphost add -c private phys-schost-1
```

You must specify the community name when you add a host to a community other than `public`.

EXAMPLE 2 Adding a Host by Specifying the Host IP and IPv6 Addresses

The following command adds a host to the SNMP host list on the current node for the community `public`. The first version of the command adds the host by specifying the IP address for the host. The second version of the command adds the host by specifying the IPv6 address for the host.

```
# clsnmphost add -c public 192.168.12.12
    or
# clsnmphost add -c public fe:1::5
```

EXAMPLE 3 Removing Hosts

The following command removes all hosts from the community `private`.

```
# clsnmphost remove -c private +
```

EXAMPLE 4 Listing Hosts on the Current Node

The following command lists all the hosts on the current node.

```
# clsnmphost list
phys-schost-1
192.168.12.12
```

EXAMPLE 5 Listing Hosts and Their Community Names

The following command uses the verbose option `-v` to list all hosts on the current node and their community names.

```
# clsnmphost list -v

--- SNMP hosts on node phys-schost-1 ---

Host Name           Community
---------           ---------
phys-schost-1       private
```

EXAMPLE 5 Listing Hosts and Their Community Names *(Continued)*

```
192.168.12.12       public
```

EXAMPLE 6 Displaying the SNMP Host Configuration

The following command displays all the configuration information for the cluster that SNMP hosts on the node `phys-cluster-2`.

```
# clsnmphost show -n phys-schost-2

--- SNMP Host Configuration on phys-schost-2 ---

SNMP Host Name:                                 phys-schost-2
  Community:                                       private
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also

clsnmpmib(1CL), cluster(1CL), Intro(1CL), sceventmib(1M), `su`(1M), scha_calls(3HA), `attributes`(5), `rbac`(5), clconfiguration(5CL)

Notes

The superuser can run all forms of this command.

All users can run this command with the `-?` (help) or `-V` (version) option.

To run the `clsnmphost` command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`add`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`remove`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`

Name clsnmpmib, clmib – administer Sun Cluster SNMP MIBs

Synopsis

```
/usr/cluster/bin/clsnmpmib  -V

/usr/cluster/bin/clsnmpmib  [subcommand] -?

/usr/cluster/bin/clsnmpmib  [subcommand] [options]
 -v [mib]

/usr/cluster/bin/clsnmpmib disable  [-n node[,...] ]
 {+ |  mib ...}

/usr/cluster/bin/clsnmpmib enable  [-n node[,...] ]
 {+ |  mib ...}

/usr/cluster/bin/clsnmpmib export  [-n node[,...] ]
 [-o  {- |  clconfigfile}] [+ |  mib ...]

/usr/cluster/bin/clsnmpmib list  [-n node[,...] ]
 [+ |  mib ...]

/usr/cluster/bin/clsnmpmib set  [-p name=value] [...]
 [-n node[,... ] {+ |  mib ...}

/usr/cluster/bin/clsnmpmib show  [-n node[,...] ]
 [+ |  mib ...]
```

Description The `clsnmpmib` command administers existing Sun Cluster Simple Network Management Protocol (SNMP) Management Information Bases (MIBs) on the current node. To create SNMP hosts that can administer the MIBs, see the clsnmphost(1CL) man page. To define SNMP Version 3 (SNMPv3) users who can access the MIBs when using the SNMPv3 protocol, see the clsnmpuser(1CL) man page.

The general form of this command is as follows:

`clsnmpmib` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the option `-?` or `-V`.

Each option of this command has a long form and a short form. Both forms of each option are provided with the description of the option in the OPTIONS section.

See the Intro(1CL) man page for more information.

Sun Cluster MIBs Sun Cluster currently supports one MIB, the Event MIB. The Sun Cluster SNMP Event MIB notifies an SNMP manager of cluster events in real time. When enabled, the Sun Cluster Event MIB automatically sends trap notifications to all hosts that are defined by the `clsnmphost` command. The Sun Cluster Event MIB sends trap notifications on port 11162. The SNMP tree is viewed on port 11161.

Because clusters generate numerous event notifications, only events with a severity of `warning` or greater are sent as trap notifications. The MIB maintains a read-only table of the most current 50 events for which a trap has been sent. This information does not persist across reboots.

You can use this command only in the global zone.

Subcommands The following subcommands are supported:

`disable`
Disables one or more of the cluster MIBs on the specified nodes.

You can use this subcommand only in the global zone.

If you do not specify the `-n` option, only MIBs on the current node are disabled. When a MIB is disabled, you cannot access the MIB tables, and the MIB does not send any trap notifications.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

`enable`
Enables one or more cluster MIBs on the specified node.

You can use this subcommand only in the global zone.

If you do not specify the `-n` option, only MIBs on the current node are enabled. To limit the MIBs that are enabled, use the *mib* operand.

When you enable a MIB, you enable all of its functionality. However, some further configuration might be necessary for all of the MIB features to be fully functional. For example, the MIB cannot send trap notifications if no hosts have been configured. For information about configuring the SNMP host, see the `clsnmphost`(1CL) man page.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`export`
Exports the cluster MIB configuration information.

You can use this subcommand only in the global zone.

Use the `-n` option to specify one or more nodes from which to export the MIB configuration information. If you use `export` without the `-n` option, the subcommand exports only MIB configuration information from the current node. By default, this subcommand exports configuration information from all MIBs on the current node. To refine the output further, specify the name of one or more MIBs for which you need configuration information.

For more information about the output format from the `export` subcommand, see the clconfiguration(5CL) man page. By default all output is sent to the standard output. Use the `-o` option followed by a file name to redirect the output to the file.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`list`
Displays a list of cluster MIBs on the specified nodes.

You can use this subcommand only in the global zone.

Use the `-n` option to specify the nodes for the cluster MIBs that you want to list. If you use the `list` subcommand without the `-n` option, the subcommand lists only the MIBs on the current node. To limit the MIBs that are listed, specify the name of one or more MIBs that you want to list.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`set`
Changes the SNMP protocol setting that is used on one or more of the MIBs on the specified nodes.

You can use this subcommand only in the global zone.

By default, this subcommand changes all MIBs on the nodes. If you do not specify the node, only the SNMP protocol for the MIBs on the current node is modified. You must specify the SNMP protocol by using the `-p` option. All MIBs use SNMPv2 as the default protocol setting. The `set` subcommand changes the protocol setting of all the MIBs, unless you use the *mib* operand to specify MIB names.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`show`
Displays information for MIBs on the specified nodes.

You can use this subcommand only in the global zone.

The `show` subcommand displays the name of the MIB and its SNMP protocol version. By default, this subcommand shows information for all MIBs on the nodes.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

Options The following options are supported:

`-?`
`--help`
Displays help information.

You can specify this option with or without a subcommand.

- If you use this option without a subcommand, the list of available subcommands is displayed.
- If you use this option with a subcommand, the usage options for that subcommand are displayed.

When you use this option, no other processing is performed.

`-n` *node*[,…]
`--node[s]` *node*`-[,…]`
Specifies a node or a list of nodes. You can specify each node as a node name or a node ID. All forms of the `clsnmpmib` command accept this option. You can use the `-n` option to specify on which node[s] you want the action to be performed. Without the `-n` option, the command assumes the current node.

`-o` {`-` | *clconfigfile*}
`--output {-` | *clconfigfile*`-}`
Specifies the location where information about the cluster MIB configuration is to be written. This location can be a file or the standard output. To specify the standard output, specify the minus sign (`-`) instead of a file name. If you specify the standard output, all other standard output for the command is suppressed. If you do not specify the `-o` option , the output is sent to the standard output. You can specify this option only with the `export` subcommand.

Configuration information is written in the format that is defined in the clconfiguration(5CL) man page.

`-p` *name=value*
`--property=`*name-=value*
`--property` *name-=value*
Specifies the version of the SNMP protocol to use with the MIBs. Sun Cluster supports the SNMPv2 and SNMPv3 protocol versions.

Multiple instances of `-p` *name=value* are allowed.

You can set the following property with this option:

`version`
Specifies the version of the SNMP protocol to use with the MIBs. You specify *value* as follows:

- `version=SNMPv2`
- `version=snmpv2`
- `version=2`
- `version=SNMPv3`
- `version=snmpv3`
- `version=3`

`-V`
`--version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options because they are ignored. The `-V` option displays only the version of the command. No other operations are performed.

`-v`
`--verbose`
Prints verbose information to the standard output.

You can specify this option with any form of the command, although some subcommands might not produce expanded output. For example, the `export` subcommand does not produce expanded output when you specify the verbose option.

Operands The following operands are supported:

mib	Specifies the name of the MIB or MIBs to which to apply the subcommand. If you do not specify this operand, the subcommand uses the default plus sign (+), which means all MIBs. If you use the *mib* operand, specify the MIB in a space-delimited list after all other command-line options.
+	All cluster MIBs.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

18 `CL_EINTERNAL`
Internal error was encountered

An internal error indicates a software defect or other defect.

35 `CL_EIO`
I/O error

A physical input/output error has occurred.

36 `CL_ENOENT`
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

Examples

EXAMPLE 1 Listing MIBs

The following command lists all MIBs on the cluster node.

```
# clsnmpmib list
Event
```

EXAMPLE 2 Enabling a MIB

The following command enables the Event MIB on the current node.

```
# clsnmpmib enable event
```

The names of cluster MIBs are not case sensitive.

EXAMPLE 3 Changing the Protocol

The following command changes the protocol of the `Event` MIB on cluster node `phys-cluster-2` to `SNMPv3`.

```
# clsnmpmib set -n phys-cluster-2 -p version=SNMPv3 Event
```

If you use the `-n` option, you can alternatively use the node ID instead of the node name.

EXAMPLE 4 Showing the Configuration

The following command displays the configuration information on cluster nodes `phys-cluster-1` and `phys-cluster-2`.

```
# clsnmpmib show -n phys-cluster-1,phys-cluster-2
--- SNMP MIB Configuration on myhost ---

SNMP MIB Name:                 phys-cluster-1
  State:                         Event
  Enabled:                       yes
  Protocol:                      SNMPv3

SNMP MIB Name:                 phys-cluster-2
  State:                         Event
  Enabled:                       yes
  Protocol:                      SNMPv3
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

Files

`/usr/cluster/lib/mib/sun-cluster-event-mib.mib`
Sun Cluster SNMP Event MIB definition file

See Also

clsnmphost(1CL), clsnmpuser(1CL), Intro(1CL), cluster(1CL), sceventmib(1M), scha_calls(3HA), `attributes`(5), `rbac`(5), clconfiguration(5CL)

Notes

The superuser can run all forms of this command.

All users can run this command with the `-?` (help) or `-V` (version) option.

To run the `clsnmpmib` command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`disable`	`solaris.cluster.modify`
`enable`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`

Subcommand	RBAC Authorization
set	solaris.cluster.modify
show	solaris.cluster.read

Name clsnmpuser – administer Sun Cluster SNMP users

Synopsis

```
/usr/cluster/bin/clsnmpuser  -V

/usr/cluster/bin/clsnmpuser  [subcommand] -?

/usr/cluster/bin/clsnmpuser  [subcommand] [options]
 -v [operand]

/usr/cluster/bin/clsnmpuser create  -i {- |  clconfigfile}
 [-a authentication] -f passwdfile [-n node[,...]]
 {+ |  user ...}

/usr/cluster/bin/clsnmpuser delete  [-a authentication]
 [-n node[,...] ] {+ |  user ...}

/usr/cluster/bin/clsnmpuser export  [-o  {- |  clconfigfile}]
 [-a authentication] [-n node[,...] ] [ {+ |  user ...}]

/usr/cluster/bin/clsnmpuser list  [-a authentication]
 [-n node[,...] ] {-d |  + |  user ...}

/usr/cluster/bin/clsnmpuser set  [-a authentication]
 [-n node[,...] ] {+ |  user ...}

/usr/cluster/bin/clsnmpuser set-default  {-l seclevel [,...] }
 {+ |  user ...}

/usr/cluster/bin/clsnmpuser show  [-a authentication]
 [-n node[,...] ] [-d |  + |  user ...]
```

Description The `clsnmpuser` command administers the roles of Simple Network Management Protocol (SNMP) users who can administer the control mechanisms on cluster Management Information Bases (MIBs). For more information about cluster MIBs, see the clsnmpmib(1CL) man page. If the cluster contains a MIB that is configured to use SNMP Version 3 (SNMPv3), you must define an SNMP user. SNMP users are not the same users as Solaris OS users, and SNMP users do not need to have the same user names as existing OS users.

This command has no short form.

The general form of this command is as follows:

`clsnmpuser` [*subcommand*] [*options*] [*operands*]

You can omit *subcommand* only if *options* specifies the option `-?` or `-V`.

Each option of this command has a long form and a short form. Both forms of each option are provided with the description of the option in the OPTIONS section.

See the Intro(1CL) man page for more information.

You can use this command only in the global zone.

Subcommands The following subcommands are supported:

`create`
Creates a user and adds the user to the SNMP user configuration on the specified node.

You can use this subcommand only in the global zone.

Use the `-n` option with this subcommand to specify the cluster node on which to create the SNMP user. If you do not specify the `-n` option, the user is created and added only to the SNMP configuration on the current node.

To create and add all of the users that are configured in the `clconfiguration` file, use the `-i` option and the `-n` option.

To assign an authentication type to the SNMP user that you are creating, specify the `-a` option.

You can include the password for the SNMP user by specifying the `-f` option. The `-f` option is required if you are using the `-i` option.

If you specify the `-i` option, the configuration information from the clconfiguration(5CL) file is used. When you specify the `-i` option, you can also specify the plus sign (+) operand or a list of users.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this command. See the `rbac`(5) man page.

`delete`
Deletes an SNMPv3 user from the specified node.

You can use this subcommand only in the global zone.

When you use the `delete` subcommand and specify only a user name, the subcommand removes all instances of the user. To delete users by authentication type, use the `-a` option. If you do not use the `-n` option, the user is deleted from only the current node.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`export`
Exports the SNMP user information from the specified node.

You can use this subcommand only in the global zone.

If you do not use the `-n` option, the SNMP user information is exported only from the current node. For the format of the output from the `export` subcommand, see the clconfiguration(5CL) man page. By default, all output is sent to standard output. Use the `-o` option followed by a file name to redirect the output to the file.

You can use the `-a` option to provide output only for those users with a specific authentication type. If you specify one or more users as operands, the output is restricted to only the information about those users.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`list`
Prints a list of SNMPv3 users that are configured on the specified node.

You can use this subcommand only in the global zone.

By default, the `list` subcommand displays all SNMPv3 users on the specified node. To display only the default SNMP user, specify the `-d` option with no operands. To restrict the output to a specified authentication type, use the `-a` option.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`set`
Changes the configuration of a user on the specified node.

You can use this subcommand only in the global zone.

If you do not specify the `-n` option, the configuration of a user is modified only on the current node.

Users other than superuser require `solaris.cluster.modify` RBAC auhorization to use this subcommand. See the `rbac`(5) man page.

`set-default`
Specifies the name of the default SNMP user and the security level that is used when a MIB sends a trap notification.

You can use this subcommand only in the global zone.

You use the `-l` option to specify the security level.

If the MIB is configured to use SNMPv3, you must specify a specific user name and security level with which to authenticate the traps. If a configuration has more than one user, you must specify the default user that the MIB will use when it sends the trap notifications.

If the configuration contains only one user, that user automatically becomes the default SNMP user. If the default SNMP user is deleted, another existing user, if any, becomes the default.

Users other than superuser require `solaris.cluster.modify` RBAC auhorization to use this subcommand. See the `rbac`(5) man page.

`show`
Prints information about the users on the specified node.

You can use this subcommand only in the global zone.

By default, the `show` subcommand displays information about all users on the node. To display information about only the default SNMP user, specify the `-d` option and do not provide an operand. To limit the output to specific authentication types, use the `-a` option. If you do not use the `-n` option, the command displays only user information from the current node.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

Options The following options are supported:

`-?`
`--help`
Prints help information.

You can specify this option with or without a subcommand.

- If you use this option without a subcommand, the list of available subcommands is displayed.
- If you use this option with a subcommand, the usage options for that subcommand are displayed.

When this option is used, no other processing is performed.

`-a` *authentication*
`--authentication` *authentication*
Specifies the authentication protocol that is used to authorize the user. The value of the authentication protocol can be `SHA` or `MD5`.

`-d`
`--default`
Specify the default SNMP user that is used when a MIB sends a trap notification.

`-f` *passwdfile*
`--file` *passwdfile*
Specifies a file that contains one or more SNMP user passwords. If you do not specify this option when you create a new user, the command prompts for a password. This option is valid only with the `create` subcommand.

User passwords must be specified on separate lines in the following format:

user:*password*

Passwords cannot contain the following characters or a space:

- ; (semicolon)
- : (colon)
- \ (backslash)
- \n (newline)

`-i` {- | *clconfigfile*}
`--input {- | clconfigfile-}`
Specifies configuration information that is to be used to validate or modify the SNMP hosts configuration. This information must conform to the format that is defined in the clconfiguration(5CL) man page. This information can be contained in a file or supplied through standard input. To specify standard input, specify the minus sign (-) instead of a file name.

`-l` *seclevel*
`--securitylevel` *seclevel*
Specifies the user's security level. You specify one of the following values for *seclevel*:

- `noAuthNoPriv`
- `AuthNoPriv`
- `authPriv`

For more information about SNMP security levels, see the `snmpcmd`(1M) man page.

`-n` *node*[,…]
`--node[s]` *node*`-[…]`
Specifies a node or a list of nodes. You can specify each node as a node name or as a node ID.

All forms of this command accept this option.

`-o` {- | *clconfigfile*}
`--output {- | clconfigfile-}`
Writes the cluster SNMP host configuration information in the format that is described by the clconfiguration(5CL) man page. This information can be written to a file or to standard output.

To write to standard output, specify the minus sign (-) instead of a file name. If you specify standard output, all other standard output for the command is suppressed.

`-V`
`--version`
Prints the version of the command.

Do not specify this option with subcommands, operands, or other options because they are ignored. The `-V` option displays only the version of the command. No other operations are performed.

`-v`
`--verbose`
Prints verbose messages and information.

You can specify this option with any form of the command, although some subcommands might not produce expanded output. For example, the `export` subcommand does not produce expanded output if you specify the verbose option.

Operands The following operands are supported:

+	Specifies all SNMP users.
user	Specifies the name of the SNMP user.

Exit Status If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`18` `CL_EINTERNAL`
Internal error was encountered

An internal error indicates a software defect or other defect.

`35` `CL_EIO`
I/O error

A physical input/output error has occurred.

`36` `CL_ENOENT`
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

Examples

EXAMPLE 1 Creating an SNMPv3 User

The following command creates a new user `newuser1` and adds the user to the configuration on the current node. The authentication type is `SHA`.

```
# clsnmpuser create -a SHA newuser1
Enter password for user 'newuser1':
```

This example requires that you enter a password for the user to be created. To automate this process, use the `-f` option.

EXAMPLE 2 Listing Users

The following command lists all users with an authentication type of `MD5`.

```
# clsnmpuser list -a MD5 +
user1
mySNMPusername
```

The plus sign (+) is optional, as it is the default.

EXAMPLE 3 Showing Users

The following command displays the user information for all users on the current node.

```
# clsnmpuser show

--- SNMP User Configuration on phys-schost-1 ---

SNMP User Name:                                 newuser1
  Authentication Protocol:                         SHA
  Default User:                                    Yes
  Default Security Level:                          authPriv
```

EXAMPLE 4 Changing a User's Authentication Protocol and Status

The following command modifies the authentication protocol and default user status of the user `newuser1`.

```
# clsnmpuser set -a MD5 newuser1
```

EXAMPLE 5 Deleting SNMP Users

The following command deletes all SNMP users.

```
# clsnmpuser delete +
```

The plus sign (+) is used in this example to specify all users.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also

clsnmphost(1CL), clsnmpmib(1CL), cluster(1CL), Intro(1CL), sceventmib(1M), `snmpcmd`(1M), `su`(1M), scha_calls(3HA), `attributes`(5), `rbac`(5), clconfiguration(5CL)

Notes

The superuser can run all forms of this command.

All users can run this command with the `-?` (help) or `-V` (version) option.

To run the `clsnmpmib` command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`set`	`solaris.cluster.modify`
`set-default`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`

Name cltelemetryattribute, clta – configure system resource monitoring

Synopsis

```
/usr/cluster/bin/cltelemetryattribute  -V

/usr/cluster/bin/cltelemetryattribute [subcommand]
 -?

/usr/cluster/bin/cltelemetryattribute subcommand
 [options] -v [telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute disable  [-i
 {- |  clconfigfile}] [-t object-type] {+ |  telemetry-attribute ...}

/usr/cluster/bin/cltelemetryattribute enable  [-i
 {- |  clconfigfile}] [-t object-type] {+ |  telemetry-attribute ...}

/usr/cluster/bin/cltelemetryattribute export  [-o
 {- |  clconfigfile}] [-t object-type[,...]]
 [+ |  telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute list  [-t object-type[,...]]
 [+ |  telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute print  [-b object-instance[,...]]
 [-a] [-d period] [-u] [-n node[,...]] [-t object-type[,...]]
 [+ |  telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute set-threshold
 -b object-instance [-n node] {-p name=value}
 [-p name=value ] [...] -t object-type telemetry-attribute

/usr/cluster/bin/cltelemetryattribute show  [-b object-instance[,...]]
 [-n node[,...]] [-t object-type[,...]] [+
 |  telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute status  -b object-instance
 [-n node] {-p name} -t object-type [+ |  telemetry-attribute ...]
```

Description This command configures the monitoring of system resources.

You can monitor the use of system resources on different types of objects, including the following:

- Disks
- File systems
- IP addresses
- Network interfaces
- Nodes
- Solaris zones
- Resource groups

The aspects of system resources that you can monitor are called telemetry attributes.

This command does the following:

- Enables or disables telemetry attributes
- Sets or modifies thresholds for telemetry attributes
- Displays a list of the attributes that are being monitored, the thresholds that are applied, and the data that is collected about objects

You select the aspects of system resource usage that you want to monitor by identifying the corresponding telemetry attribute. To monitor system resource usage on an object, you enable the corresponding telemetry attributes on that type of object. The Sun Cluster software collects usage data for these attributes on all objects of that type in the cluster.

For a system resource, a particular value might be critical for the performance of your cluster. You can set a threshold for a telemetry attribute so that you are notified if the critical value is crossed. See the `set-threshold` subcommand and the description of the `-p` option for information about thresholds.

You can omit *subcommand* only if *options* is the `-?` option or the `-V` option.

Each option has a long and a short form. Both forms of each option are given with the description of the option in OPTIONS.

The `clta` command is the short form of the `cltelemetryattribute` command.

Before you refine the configuration of system resource monitoring, you must initialize monitoring. See the sctelemetry(1M) man page.

You can use this command only in the global zone.

Subcommands The following subcommands are supported:

`disable`
Disables the specified telemetry attribute for the specified object type.

You can use this subcommand only in the global zone.

The Sun Cluster software collects usage data for system resources that are set to an `enabled` state. If you set a system resource for an object type to the `disabled` state, Sun Cluster software does not collect data for any instance that corresponds to that object instance.

The `cltelemetryattribute` command also disables the data collection on these attributes when both of the following conditions are met:

- You specify a configuration file with the `-i` option.
- The telemetry attributes are set to `disabled` in the input file.

You create the configuration file by using the `export` subcommand.

When you set a telemetry attribute to `disabled`, the settings of its configured thresholds remain unaltered.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

`enable`
Enables data collection of the specified telemetry attribute of the specified object type.

You can use this subcommand only in the global zone.

By default, selected attributes are enabled for selected object types.

To enable data collection of telemetry attributes, set the telemetry attributes to `enabled`.

The Sun Cluster software collects data on only an object type for the telemetry attributes that are enabled for that object type. When you enable an attribute for an object type, Sun Cluster software collects data for that attribute for all object instances of that type on all nodes.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`export`
Exports the configuration of the telemetry attributes of object types and object instances to a file or to the standard output (`stdout`).

You can use this subcommand only in the global zone.

The configuration includes whether an attribute is enabled or disabled for an object type. The configuration can also include the limits that are set for the threshold configuration.

Specify a file by using the `-o` option to write the configuration information to a file. If you do not specify the `-o` option, the `cltelemetryattribute` command writes the configuration information to the standard output (`stdout`).

The `export` subcommand does not modify cluster configuration data.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`list`
Displays the telemetry attributes that you can configure for the specified object types.

You can use this subcommand only in the global zone.

If you specify the verbose option `-v`, the `list` subcommand displays the type of object to which you can apply the attribute.

The properties of a threshold are displayed in the following format:

`Threshold:` *`severity`*`,` *`direction`*`,` *`value`*`,` *`rearm`*

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`print`
Displays system resource usage for the specified telemetry attributes that are enabled for the specified object instances or object types.

You can use this subcommand only in the global zone.

The output includes the following data:

- Date and timestamp
- Object instance
- Object type
- Telemetry attribute
- Node
- Value

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`set-threshold`
Modifies the settings of a threshold for a specified telemetry attribute for a specified object on a node.

You can use this subcommand only in the global zone.

Use the `-p` option to specify the threshold to be modified. Also use the `-p` option to specify the threshold properties that you want to modify. You can modify only the `value` and `rearm` threshold properties.

You must change at least one of these properties for the specified threshold.

To deactivate a threshold, specify a blank for `value` and `rearm`, as follows:

```
-y value=,rearm=
```

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`show`
Displays the properties that are configured for telemetry attributes on object types or object instances.

You can use this subcommand only in the global zone.

These attributes include whether the system resources are enabled for an object type. If you specify the verbose option `-v`, the `show` subcommand displays the threshold settings for the telemetry attributes that are enabled for object instances.

The properties of a threshold are displayed in the following format:

`Threshold:` *`severity`*`,` *`direction`*`,` *`value`*`,` *`rearm`*

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`status`
Displays the current status of object types on which thresholds are configured on the standard output.

You can use this subcommand only in the global zone.

Use this subcommand without arguments to display the status for all active thresholds that currently have a warning or a fatal severity level. Possible output for thresholds includes the current severity level of the thresholds.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

Options The following options are supported:

`-?`
`--help`
Displays help information.

You can specify this option with or without a *subcommand*.

If you specify this option without a *subcommand*, the list of all available subcommands is displayed.

If you specify this option with a *subcommand*, the usage for that *subcommand* is displayed.

If you specify this option with the `set-threshold` subcommand, help information is displayed for all resource group properties.

If you specify this option with other options, with subcommands, or with operands, they are all ignored. No other processing occurs.

`-a`
`--average`
Prints the average of the data collected over a three-hour period and the standard deviation that is associated with this average.

If you specify the `-a` option and the `-d` option together, data that is averaged over three-hour intervals within the specified *period* is printed.

If you do not specify the `-a` option, the data that is printed is the latest data.

`-b` *object-instance*
`--object-instance=`*object-instance*
`--object-instance` *object-instance*
Specifies an object instance about which you want to display information or for which you want to set a threshold.

An object instance is always of a certain type. For example, a cluster node `phys-schost-1` is an object instance of type `node`. The Sun Cluster software monitors system resources of an object instance only if the corresponding telemetry attributes are enabled for the object type.

`-d` *period*
`--date-range=`*period*
`--date-range` *period*
Specifies the period during which you want the Sun Cluster software to collect monitoring data.

The format of the dates and times that you specify for the *period* argument must conform to the International Organization for Standardization (ISO) 8601 International Date Format.

begin-time`,`*end-time*]	The period is between the two times that are separated by the comma (`,`).
begin-time`+`	The period is between the specified begin time and the current time.
end-time`-`	The period is between the time that the Sun Cluster software starts and begins collecting data and the specified end time.

Examples of the format of *period* are as follows:

`-d` **`2006-04-30T18:00,2006-06-16T18:00`**
From 6:00 p.m. on 30 April 2006 to 6:00 p.m. on 16 June 2006

`-d` **`2006-06-16+`**
From 12:00 midnight on 16 June 2006 onwards

`-d` **`2006-07-31T18:00+`**
From 6:00 p.m. on 31 July 2006 onwards

`-d` **`2006-06-16T18:00–`**
From the time that the Sun Cluster software starts to 6:00 p.m. on 16 June 2006

`-d` **`2006-05-31T12:00,2006-06-16T11:59`**
From 12:00 midnight on 31 May 2006 to 11:59 p.m. on 16 June 2006

You can use this option only with the `print` subcommand.

`-i` {`-` | *clconfigfile*}
`--input={- |` *clconfigfile*`-}`
`--input {- |` *clconfigfile*`-}`
Specifies that you want to use the configuration information that is located in the *clconfigfile* file to specify the telemetry attribute and threshold configuration. See the clconfiguration(5CL) man page.

Specify a dash (`-`) with this option to provide configuration information through the standard input (`stdin`). If you specify other options, they take precedence over the options and information in *clconfigfile*.

`-n` *node*
`--node=`*node*
`--node` *node*
Specifies the node name on which Sun Cluster collects usage data. You can specify a name or a node identifier.

Do not use the `-n` option when specifying subcommands on object instances of type `node`, a resource, or a resource group.

`-o` {`-` | *clconfigfile*}
`--output={-` | *clconfigfile*`-}`
`--output {-` | *clconfigfile*`-}`
Writes the telemetry attribute and threshold configuration data to a file or to the standard output (`stdout`). The format of the configuration information is described in the clconfiguration(5CL) man page.

If you specify a file name with this option, this option creates a new file. Configuration information is then placed in that file. If you specify `-` with this option, the configuration information is sent to the standard output (`stdout`). All other standard output for the command is suppressed.

You can use this option only with the `export` subcommand.

`-p` *name*
`--property=`*name*
`--property` *name*
Specifies a list of properties for the `status` subcommand.

For information about the properties for which you can set thresholds with the `set-threshold` subcommand, see the description of the `-p` *name*=*value* option.

`-p` *name*=*value*
`--property=`*name*`-=`*value*
`--property` *name*`-` *value*
Specifies the properties of a threshold.

Multiple instances of `-p` *name*=*value* are allowed.

For information about the properties about which you can display information with the `status` subcommand, see the description of the `-p` *name* option.

For each threshold, you must specify a `severity` property and a `direction` property to identify the threshold. You cannot modify these properties after a threshold has been set.

Set a `value` for each threshold. You can also set a `rearm` for each threshold. Use the `set-threshold` subcommand to modify the `value` and `rearm` properties. Properties and values that you can specify with this option are as follows:

`severity`
The severity level of a threshold. The possible values to which you can set this property are `fatal` and `warning`. A threshold with a severity level of `fatal` is more critical than a threshold with a severity level of `warning`.

The severity level is displayed as a visual alarm in Sun Cluster Manager.

`direction`
The direction of the threshold that is to be applied. The possible values to which you can set this property are `falling` and `rising`. By setting the `direction` property to `falling`, you specify that the `fatal` severity level has a lower value than the `warning` severity level. By setting the `direction` property to `rising`, you specify that the `fatal` severity level has a higher value than the `warning` severity level.

`value`
The value for which you want to set a threshold on a telemetry attribute. If the threshold value is crossed, the severity of the telemetry attribute changes. You can associate up to four thresholds with a particular telemetry attribute on an object.

Use the `set-threshold` subcommand to set or modify the `value` property.

`rearm`
A means of clearing the severity on a telemetry attribute. By specifying a rearm value, the severity on a telemetry attribute is cleared when the value of the telemetry attribute crosses the rearm value in the direction opposed to that set in the `direction` property. If you do not specify the rearm value, the rearm value is as if the threshold value and the rearm value are set to the same value.

The frequency of notifications follows the principle of *hysteresis*, that is, the frequency is determined by a double-valued function. One value applies when the function is increasing. The other value applies when the function is the same as the value.

Set the values of `rearm` and `value` to suit your system. If you do not specify the optional `rearm` property, it takes `value` as the default. However, if you set the `rearm` property to the same value as the `value` property, or if you do not assign a value to `rearm`, you receive a notification every time that the value of the monitored telemetry attribute goes above or below the value that is set for `value`. To avoid receiving a large number of notifications, set `rearm` to a value other than `value`.

If you specify `rearm` with the `set-threshold` subcommand, the `cltelemetryattribute` command ensures that the value of `rearm` complies with the following requirements:

- If `direction` is `rising`, `value` has a value that is greater than or equal to the `rearm`.

- If `direction` is `falling`, `value` has a value that is smaller than or equal to `value`.

Use the `set-threshold` subcommand to change the `rearm`.

`-t` *object-type*
`--object-type=`*object-type*
`--object-type` *object-type*
Specifies the type of object on which the Sun Cluster software is to collect usage data. All object instances are of a certain type.

Use this option to limit the output of subcommands to objects of the specified type.

The object types for which you can monitor system resources and each object type's associated telemetry attributes are as follows:

Object Type	Description	Telemetry Attribute
`disk`	Disk	`rbyte.rate`, `wbyte.rate`, `read.rate`, `write.rate`,
`filesystem`	File system	`block.used`, `inode.used`
`ipaddr`	IP address	`ipacket.rate`, `opacket.rate`
`netif`	Network interface	`ipacket.rate`, `opacket.rate`, `rbyte.rate`, `wbyte.rate`
`node`	Node	`cpu.idle`, `cpu.iowait`, `cpu.used`, `loadavg.1mn`, `loadavg.5mn`, `loadavg.15mn`, `mem.used`, `mem.free`, `swap.used`, `swap.free`
`resourcegroup`	Resource group	`cpu.used`, `mem.used`, `swap.used`
`zone`	Zone	`cpu.idle`, `cpu.iowait`, `cpu.used`, `loadavg.1mn`, `loadavg.5mn`, `loadavg.15mn`

The telemetry attributes that you can monitor are as follows:

Telemetry Attribute	Description
`block.used`	Percentage of blocks that are used on a device
`cpu.idle`	Amount of free CPU
`cpu.iowait`	Amount of CPU waiting for input/output completion
`cpu.used`	Amount of CPU that is used

Telemetry Attribute	Description
`inode.used`	Percentage of inodes that are used on a device
`ipacket.rate`	Number of incoming packets per second
`loadavg.1mn`	Number of processes that waited for the CPU in the last minute
`loadavg.5mn`	Number of processes that waited for the CPU in the last five minutes
`loadavg.15mn`	Number of processes that waited for the CPU in the last fifteen minutes
`mem.free`	Number of Mbytes of free memory
`mem.used`	Number of Mbytes of memory that is used
`opacket.rate`	Number of outgoing packets per second
`rbyte.rate`	Number of Mbits that are read per second
`read.rate`	Number of read operations per second
`swap.free`	Number of Mbytes of free swap memory
`swap.used`	Number of Mbytes of swap memory that is used
`wbyte.rate`	Number of Mbits that are written per second
`write.rate`	Number of write operations per second

You cannot monitor all telemetry attributes that are listed in the preceding table for all object types. Use the `list` subcommand to display object types on which you can collect data, and telemetry attributes that you can monitor on each type of object.

`-u`
`--utc`
Display the date and time that is shown with usage data in Coordinated Universal Time (UTC) or in Greenwich Mean Time (GMT). By specifying this option, you bypass the conversion of the date and time to, or from, the local date and time. By default, Sun Cluster software displays the local date and time.

You can use this option only with the `print` subcommand.

`-v`
`--verbose`
Displays verbose information on the standard output (`stdout`).

`-V`
`--version`
Displays the version of the command.

If you specify this option with other options, with subcommands, or with operands, they are all ignored. Only the version of the command is displayed. No other processing occurs.

Operands The following operands are supported:

telemetry-attribute Particular telemetry attribute about which you want usage data.

The Sun Cluster software contains particular types of objects on which you can collect usage data. For each object type, you can enable monitoring of telemetry attributes. The Sun Cluster software only collects data for attributes that are enabled.

\+ All telemetry groups.

Exit Status The complete set of exit status codes for all commands in this command set are listed on the Intro(1CL) man page.

If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0` `CL_NOERR`
No error

The command that you issued completed successfully.

`1` `CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3` `CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

`6` `CL_EACCESS`
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the `su`(1M) and `rbac`(5) man pages for more information.

`18` `CL_EINTERNAL`
Internal error was encountered

An internal error indicates a software defect or other defect.

`35` `CL_EIO`
I/O error

A physical input/output error has occurred.

36 CL_ENOENT
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the `-o` option does not exist.
- The configuration file that you attempted to access with the `-i` option contains errors.

38 CL_EBUSY
Object busy

You attempted to remove a cable from the last cluster interconnect path to an active cluster node. Or, you attempted to remove a node from a cluster configuration from which you have not removed references.

39 CL_EEXIST
Object exists

The device, device group, cluster interconnect component, node, cluster, resource, resource type, or resource group that you specified already exists.

41 CL_ETYPE
Unknown type

The type that you specified with the `-t` or `-p` option does not exist.

Examples

EXAMPLE 1 Displaying System Resources That Are Configured for an Object Type

The following command displays the system resources that are applicable to an object type, in this case a disk.

```
# cltelemetryattribute list -t disk
rbyte.rate
wbyte.rate
write.rate
read.rate
```

EXAMPLE 2 Enabling Telemetry Attributes for an Object Type

The following command enables data collection for the specified telemetry attributes on all disks in the cluster.

```
# cltelemetryattribute enable -t disk rbyte.rate wbyte.rate
```

EXAMPLE 3 Setting a Threshold for a Telemetry Attribute of an Object Type

The following command sets a threshold for the telemetry attribute `wbyte.rate` on disk `d4` in the cluster. The default value of `rearm` is set to the value of `value`. Consequently, when the number of bytes that are written to disk `d4` exceeds or falls below 100, the Sun Cluster software issues a fatal notification.

```
# cltelemetryattribute set-threshold -t disk -b d4 \
-p severity=fatal,direction=rising,value=100 wbyte.rate
```

EXAMPLE 4 Showing the Non-Verbose List of Configured Telemetry Attributes

The following command shows the non-verbose list of telemetry attributes that are configured on all the disks in a cluster.

```
# cltelemetryattribute show -t disk

=== Telemetry Attributes ===

Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk
```

EXAMPLE 5 Showing the Verbose List of Configuration of Telemetry Attributes

The following command shows the verbose list of telemetry attributes that are configured on all the disks in the cluster.

```
# cltelemetryattribute show -v -t disk

=== Telemetry Attributes ===

Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk
```

EXAMPLE 5 Showing the Verbose List of Configuration of Telemetry Attributes *(Continued)*

```
Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

% cltelemetryattribute show -v -t disk

=== Telemetry Attributes ===

Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk

  --- Object Instances of Type "disk" ---

  Object Instance:                              d4
    Thresholds:                                 <Direction, Severity, Value, Rearm>
      Threshold 1:                                 <rising, fatal, 1000, 500>

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk
```

EXAMPLE 6 Showing the Status of Telemetry Attributes

The following command shows the status of telemetry attributes that are configured on all the disks in the cluster.

```
# cltelemetryattribute status
```

EXAMPLE 6 Showing the Status of Telemetry Attributes *(Continued)*

```
=== Telemetry Attributes Thresholds ===

Attribute   Obj-Instance   Obt-Type   Node      Threshold                      Status
Status
---------   ------------   --------   -------   ---------------------------   -------
mem.used    phys-schost-1  node       16-v2-4   <rising, fatal, 1000, 1000>   warning
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Interface Stability	Evolving

See Also

cluster(1CL), Intro(1CL), sctelemetry(1M), `su`(1M), `attributes`(5), `rbac`(5), SUNW.SCTelemetry(5), clconfiguration(5CL)

Notes

The superuser can run all forms of this command.

All users can run this command with the `-?` (help) or `-V` (version) option.

To run the `cltelemetryattribute` command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`disable`	`solaris.cluster.modify`
`enable`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`print`	`solaris.cluster.read`
`set-threshold`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`
`status`	`solaris.cluster.read`

Name cluster – manage the global configuration and status of a cluster

Synopsis

```
/usr/cluster/bin/cluster  -V

/usr/cluster/bin/cluster [subcommand]
 -?

/usr/cluster/bin/cluster subcommand
 [options] -v [clustername ...]

/usr/cluster/bin/cluster create  -i {- |  clconfigfile}
 [clustername]

/usr/cluster/bin/cluster export  [-o  {- |  configfile}]
 [-t objecttype[,...]] [clustername]

/usr/cluster/bin/cluster list  [clustername]

/usr/cluster/bin/cluster list-cmds  [clustername]

/usr/cluster/bin/cluster rename  -c newclustername
 [clustername]

/usr/cluster/bin/cluster restore-netprops  [clustername]

/usr/cluster/bin/cluster set  {-p name=value} [-p name=value ]
 [...] [clustername]

/usr/cluster/bin/cluster set-netprops  {-p name=value}
 [-p name=value ] [...] [clustername]

/usr/cluster/bin/cluster show  [-t objecttype[,...]]
 [clustername]

/usr/cluster/bin/cluster show-netprops  [clustername]

/usr/cluster/bin/cluster shutdown  [-y] [-g graceperiod]
 [-m message] [clustername]

/usr/cluster/bin/cluster status  [-t objecttype[,...]]
 [clustername]
```

Description The `cluster` command displays and manages cluster-wide configuration and status information. This command also shuts down a cluster.

Almost all subcommands that you use with the `cluster` command operate in cluster mode. You can run these subcommands from any node in the cluster. However, the `create`, `set-netprops`, and `restore-netprops` subcommands are an exception. You must run these subcommands in noncluster mode.

You can omit *subcommand* only if *options* is the `-?` option or the `-V` option.

The `cluster` command does not have a short form.

Each option has a long and a short form. Both forms of each option are given with the description of the option in OPTIONS.

You can use some forms of this command in a non-global zone, referred to simply as a zone. For more information about valid uses of this command in zones, see the descriptions of the individual subcommands. For ease of administration, use this command in the global zone.

Subcommands The following subcommands are supported:

`create`
Creates a new cluster by using configuration information that is stored in a *clconfigfile* file.

The format of this configuration information is described in the clconfiguration(5CL) man page.

You can use this subcommand only in the global zone.

You must run this subcommand in noncluster mode. You must also run this subcommand from a host that is not already configured as part of a cluster. Sun Cluster software must already be installed on every node that is going to be a part of the cluster.

If you do not specify a cluster name, the name of the cluster is taken from the *clconfigfile* file.

Users other than superuser require `solaris.cluster.modify` role-based access control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

`export`
Exports the configuration information.

You can use this subcommand only in the global zone.

If you specify a file with the `-o` option, the configuration information is written to that file. If you do not specify the `-o` option, the output is written to the standard output (`stdout`).

The following option limits the information that is exported:

`-t` *objecttype*[,…] — Exports configuration information only for components that are of the specified types.

You can export configuration information only for the cluster on which you issue the `cluster` command. If you specify the name of a cluster other than the one on which you issue the `cluster` command, this subcommand fails.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`list`
Displays the name of the cluster.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`list-cmds`
Prints a list of all available Sun Cluster commands.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`rename`
Renames the cluster.

You can use this subcommand only in the global zone.

Use the `-c` option with this subcommand to specify a new name for the cluster.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`restore-netprops`
Resets the cluster private network settings of the cluster.

You can use this subcommand only in the global zone.

You must run this subcommand in noncluster mode.

Use this subcommand only when the `set-netprops` subcommand fails and the following conditions exist:

- You are attempting to modify the private network properties.
- The failure indicates an inconsistent cluster configuration on the nodes. In this situation, you need to run the `restore-netprops` subcommand.

You must run this subcommand on every node in the cluster. This subcommand repairs the cluster configuration. This subcommand also removes inconsistencies that are caused by the failure of the modification of the IP address range. In case of a failure, any attempts that you make to change the configuration settings are not guaranteed to work.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`set`
Modifies the properties of the cluster.

You can use this subcommand only in the global zone.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`set-netprops`
Modifies the private network properties.

You can use this subcommand only in the global zone.

You must run this subcommand in noncluster mode.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`show`
Displays detailed configuration information about cluster components.

You can use this subcommand only in the global zone.

The following option limits the information that is displayed:

`-t` *objecttype*[,…] — Displays configuration information only for components that are of the specified types.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`show-netprops`
Displays information about the private network properties of the cluster.

You can use this subcommand only in the global zone.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`shutdown`
Shuts down the cluster in an orderly fashion.

You can use this subcommand only in the global zone.

If you provide the name of a cluster other than the cluster on which you issue the `cluster` command, this subcommand fails.

Run this subcommand from only one node in the cluster.

This subcommand performs the following actions:

- Takes offline all functioning resource groups in the cluster. If any transitions fail, this subcommand does not complete and displays an error message.
- Unmounts all cluster file systems. If an unmount fails, this subcommand does not complete and displays an error message.
- Shuts down all active device services. If any transition of a device fails, this subcommand does not complete and displays an error message.

- Halts all nodes.

Before this subcommand starts to shut down the cluster, it issues a warning message on all nodes. After issuing the warning, this subcommand issues a final message that prompts you to confirm that you want to shut down the cluster. To prevent this final message from being issued, use the `-y` option.

By default, the `shutdown` subcommand waits 60 seconds before it shuts down the cluster. You can use the `-g` option to specify a different delay time.

To specify a message string to appear with the warning, use the `-m` option.

Users other than superuser require `solaris.cluster.admin` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

`status`
Displays the status of cluster components.

You can use this subcommand in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

The option `-t` *objecttype*[,…] displays status information for components that are of the specified types only.

Users other than superuser require `solaris.cluster.read` RBAC authorization to use this subcommand. See the `rbac`(5) man page.

Options The following options are supported:

Note – Both the short and the long form of each option are shown in this section.

`-?`
`--help`
Displays help information.

You can specify this option with or without a *subcommand*.

If you do not specify a *subcommand*, the list of all available subcommands is displayed.

If you specify a *subcommand*, the usage for that subcommand is displayed.

If you specify this option and other options, the other options are ignored.

`-c` *newclustername*
`--newclustername=`*newclustername*
`--newclustername` *newclustername*
Specifies a new name for the cluster.

Use this option with the `rename` subcommand to change the name of the cluster.

`-g` *graceperiod*
`--graceperiod=`*graceperiod*
`--graceperiod` *graceperiod*
Changes the length of time before the cluster is shut down from the default setting of 60 seconds.

You specify *graceperiod* in seconds.

`-i` {`-` | *clconfigfile*}
`--input={- |` *clconfigfile*`-}`
`--input {- |` *clconfigfile*`-}`
Uses the configuration information in the *clconfigfile* file. See the clconfiguration(5CL) man page.

To provide configuration information through the standard input (`stdin`), specify a dash (`-`) with this option.

If you specify other options, they take precedence over the options and information in the cluster configuration file.

`-m` *message*
`--message=`*message*
`--message` *message*
Specifies a message string that you want to display with the warning that is displayed when you issue the `shutdown` subcommand.

The standard warning message is `system will be shut down in ...`.

If *message* contains more than one word, delimit it with single (') quotation marks or double (") quotation marks. The `shutdown` command issues messages at 7200, 3600, 1800, 1200, 600, 300, 120, 60, and 30 seconds before a shutdown begins.

`-o` {`-` | *clconfigfile*}
`--output={- |` *clconfigfile*`-}`
`--output {- |` *clconfigfile*`-}`
Writes cluster configuration information to a file or to the standard output (`stdout`). The format of the configuration information is described in the clconfiguration(5CL) man page.

If you specify a file name with this option, this option creates a new file. Configuration information is then placed in that file. If you specify `-` with this option, the configuration information is sent to the standard output (`stdout`). All other standard output for the command is suppressed.

You can use this option only with the `export` subcommand.

`-p` *name=value*
`--property=`*name-=value*
`--property` *name-=value*

Modifies cluster-wide properties.

Multiple instances of `-p` *name=value* are allowed.

Use this option with the `set` and the `set-netprops` subcommands to modify the following properties:

`installmode`
Specify the installation-mode setting for the cluster. You can specify either `enabled` or `disabled` for the `installmode` property.

While the `installmode` property is enabled, nodes do not attempt to reset their quorum configurations at boot time. Also, while in this mode, many administrative functions are blocked. When you first install a cluster, the `installmode` property is enabled.

After all nodes have joined the cluster for the first time, and shared quorum devices have been added to the configuration, you must explicitly disable the `installmode` property. When you disable the `installmode` property, the quorum vote counts are set to default values. If quorum is automatically configured during cluster creation, the `installmode` property is disabled as well after quorum has been configured.

`heartbeat_quantum`
Define how often to send heartbeats, in milliseconds.

Sun Cluster software uses a 1 second, or 1,000 milliseconds, heartbeat quantum by default. Specify a value between 100 milliseconds and 10,000 milliseconds.

`heartbeat_timeout`
Define the time interval, in milliseconds, after which, if no heartbeats are received from the peer nodes, the corresponding path is declared as down.

Sun Cluster software uses a 10 second, or 10,000 millisecond, heartbeat timeout by default. Specify a value between 2,500 milliseconds and 60,000 milliseconds.

The `set` subcommand allows you to modify the global heartbeat parameters of a cluster, across all the adapters.

Sun Cluster software relies on heartbeats over the private interconnect to detect communication failures among cluster nodes. If you reduce the heartbeat timeout, Sun Cluster software can detect failures more quickly. The time that is required to detect failures decreases when you decrease the values of heartbeat timeout. Thus, Sun Cluster software recovers more quickly from failures. Faster recovery increases the availability of your cluster.

Even under ideal conditions, when you reduce the values of heartbeat parameters by using the `set` subcommand, there is always a risk that spurious path timeouts and node panics might occur. Always test and thoroughly qualify the lower values of heartbeat parameters under relevant workload conditions before actually implementing them in your cluster.

The value that you specify for `heartbeat_timeout` must always be greater than or equal to five times the value that you specify for `heartbeat_quantum` (`heartbeat_timeout` >= (5*`heartbeat_quantum`)).

`global_fencing`
Specify the global default fencing algorithm for all shared devices.

Acceptable values for this property are `prefer3` and `pathcount`. The `pathcount` setting determines the fencing protocol by the number of DID paths that are attached to the shared device. For devices that use three or more DID paths, this property is set to the SCSI-3 protocol.

By default, this property is set to `pathcount`.

Private network properties
You modify private network properties with the `set-netprops` subcommand only.

You must modify these private network settings only if the default private network address collides with an address that is already in use. You must also modify these private network settings if the existing address range is not sufficient to accommodate the growing cluster configuration.

All nodes of the cluster are expected to be available and in noncluster mode when you modify network properties. You modify the private network settings on only one node of the cluster, as the settings are propagated to all nodes.

When you set the `private_netaddr` property, you can also set the `private_netmask` property or the `max_nodes` and `max_privatenets` properties, or all properties. If you attempt to set the `private_netmask` property and either the `max_nodes` or the `max_privatenets` property, an error occurs. You must always set both the `max_nodes` or the `max_privatenets` properties together.

The default private network address is `172.16.0.0`, with a default netmask of `255.255.248.0`.

If you fail to set a property due to an inconsistent cluster configuration, run the `cluster restore-netprops` command on each node in noncluster mode.

Private network properties are as follows:

`private_netaddr`
Specify the private network address.

`private_netmask`
Specify the cluster private network mask. The value that you specify in this case must be equal to or greater than the default netmask `255.255.248.0`. You can set this property only in conjunction with the `private_netaddr` property.

If you want to assign a smaller IP address range than the default, you can use the `max_nodes` and `max_privatenets` properties instead of or in addition to the `private_netmask` property.

`max_nodes`
Specify the maximum number of nodes that you expect to be a part of the cluster. Include in this number the expected number of non-global zones that will use the private network. You can set this property only in conjunction with the `private_netaddr` and `max_privatenets` properties, and optionally with the `private_netmask` property. The maximum value for `max_nodes` is 64. The minimum value is 2.

`max_privatenets`
Specify the maximum number of private networks that you expect to be used in the cluster. You can set this property only in conjunction with the `private_netaddr` and `max_nodes` properties, and optionally with the `private_netmask` property. The maximum value for `max_privatenets` is 128. The minimum value is 2.

The command performs the following tasks for each combination of private network properties:

`-p private_netaddr=`*netaddr*
The command assigns the default netmask, `255.255.248.0`, to the private interconnect. The default IP address range accommodates a maximum of 64 nodes and 10 private networks.

`-p private_netaddr=`*netaddr*`,private_netmask=`*netmask*
If the specified netmask is less than the default netmask, the command fails and exits with an error.

If the specified netmask is equal to or greater than the default netmask, the command assigns the specified netmask to the private interconnect. The resulting IP address range accommodates a maximum of 64 nodes and 10 private networks.

To assign a smaller IP address range than the default, specify the `max_nodes` and `max_privatenets` properties instead of or in addition to the `private_netmask` property.

`-p private_netaddr=`*netaddr*`,max_nodes=`*nodes*`,`
`max_privatenets=`*privatenets*
The command calculates the minimum netmask to support the specified number of nodes and private networks. The command then assigns the calculated netmask to the private interconnect.

`-p private_netaddr=`*netaddr*`,private_netmask=`*netmask*`,`
`max_nodes=`*nodes*`,max_privatenets=`*privatenets*
The command calculates the minimum netmask that supports the specified number of nodes and private networks.

The command compares that calculation to the specified netmask. If the specified netmask is less than the calculated netmask, the command fails and exits with an error. If the specified netmask is equal to or greater than the calculated netmask, the command assigns the specified netmask to the private interconnect.

`-t` *objecttype*[,...]
`--type=`*objecttype*`-[,…]`
`--type` *objecttype*`-[,…]`

Specifies object types for the `export`, `show`, and `status` subcommands.

Use this option to limit the output of the `export`, `show`, and `status` subcommands to objects of the specified type only. The following object or component types are supported. Note that the status is not available for some of the object types.

Object Type	Short Object Type	Available Status
`access`	`access`	No
`device`	`dev`	Yes
`devicegroup`	`dg`	Yes
`global`	`global`	No
`interconnect`	`intr`	Yes
`nasdevice`	`nas`	No
`node`	`node`	Yes
`quorum`	`quorum`	Yes
`reslogicalhostname`	`rslh`	Yes
`resource`	`rs`	Yes
`resourcegroup`	`rg`	Yes
`resourcetype`	`rt`	No
`ressharedaddress`	`rssa`	Yes
`snmphost`	`snmphost`	No
`snmpmib`	`snmpmib`	No
`snmpuser`	`snmpuser`	No
`telemetryattribute`	`ta`	No

`-v`
`--verbose`

Displays verbose information on the standard output (`stdout`).

`-V`
`--version`
Displays the version of the command.

If you specify this option with other options, with subcommands, or with operands, they are all ignored. Only the version of the command is displayed. No other processing occurs.

`-y`
`--yes`
Prevents the prompt that asks you to confirm a shutdown from being issued. The cluster is shut down immediately, without user intervention.

Operands The following operands are supported:

clustername The name of the cluster that you want to manage.

For all subcommands except `create`, the *clustername* that you specify must match the name of the cluster on which you issue the `cluster` command.

You specify a new and a unique cluster name by using the `create` subcommand.

Exit Status The complete set of exit status codes for all commands in this command set are listed in the Intro(1CL) man page. Returned exit codes are also compatible with the return codes that are described in the scha_calls(3HA) man page.

If the command is successful for all specified operands, it returns zero (`CL_NOERR`). If an error occurs for an operand, the command processes the next operand in the operand list. The returned exit code always reflects the error that occurred first.

This command returns the following exit status codes:

`0 CL_NOERR`
No error

The command that you issued completed successfully.

`1 CL_ENOMEM`
Not enough swap space

A cluster node ran out of swap memory or ran out of other operating system resources.

`3 CL_EINVAL`
Invalid argument

You typed the command incorrectly, or the syntax of the cluster configuration information that you supplied with the `-i` option was incorrect.

6 CL_EACCESS
Permission denied

The object that you specified is inaccessible. You might need superuser or RBAC access to issue the command. See the su(1M) and rbac(5) man pages for more information.

35 CL_EIO
I/O error

A physical input/output error has occurred.

36 CL_ENOENT
No such object

The object that you specified cannot be found for one of the following reasons:

- The object does not exist.
- A directory in the path to the configuration file that you attempted to create with the -o option does not exist.
- The configuration file that you attempted to access with the -i option contains errors.

Examples

EXAMPLE 1 Displaying Cluster Configuration Information

The following command displays all available configuration information for the cluster.

```
# cluster show
    Enabled:                                        yes
    privatehostname:                                clusternode1-priv
    reboot_on_path_failure:                         disabled
    globalzoneshares:                               1
    defaultpsetmin:                                 1
    quorum_vote:                                    1
    quorum_defaultvote:                             1
    quorum_resv_key:                                0x441699B200000001
    Transport Adapter List:                         hme1, qfe3
    Node Zones:                                     phys-schost-1:za

    --- Transport Adapters for phys-schost-1 ---

    Transport Adapter:                           hme1
      Adapter State:                                Enabled
      Adapter Transport Type:                       dlpi
      Adapter Property(device_name):                hme
      Adapter Property(device_instance):            1
      Adapter Property(lazy_free):                  0
      Adapter Property(dlpi_heartbeat_timeout):     10000
      Adapter Property(dlpi_heartbeat_quantum):     1000
      Adapter Property(nw_bandwidth):               80
      Adapter Property(bandwidth):                  10
```

EXAMPLE 1 Displaying Cluster Configuration Information *(Continued)*

```
    Adapter Property(ip_address):                172.16.0.129
    Adapter Property(netmask):                   255.255.255.128
    Adapter Port Names:                          0
    Adapter Port State(0):                       Enabled

  Transport Adapter:                           qfe3
    Adapter State:                               Enabled
    Adapter Transport Type:                      dlpi
    Adapter Property(device_name):               qfe
    Adapter Property(device_instance):           3
    Adapter Property(lazy_free):                 1
    Adapter Property(dlpi_heartbeat_timeout):    10000
    Adapter Property(dlpi_heartbeat_quantum):    1000
    Adapter Property(nw_bandwidth):              80
    Adapter Property(bandwidth):                 10
    Adapter Property(ip_address):                172.16.1.1
    Adapter Property(netmask):                   255.255.255.128
    Adapter Port Names:                          0
    Adapter Port State(0):                       Enabled

  --- SNMP MIB Configuration on phys-schost-1 ---

  SNMP MIB Name:                               Event
    State:                                       SNMPv

  --- SNMP Host Configuration on phys-schost-1 ---

  --- SNMP User Configuration on phys-schost-1 ---

Node Name:                                     phys-schost-2
  Node ID:                                     2
  Type:                                        cluster
  Enabled:                                     yes
  privatehostname:                             clusternode2-priv
  reboot_on_path_failure:                      disabled
  globalzoneshares:                            1
  defaultpsetmin:                              1
  quorum_vote:                                 1
  quorum_defaultvote:                          1
  quorum_resv_key:                             0x441699B200000002
  Transport Adapter List:                      hme1, qfe3
  Node Zones:                                  phys-schost-2:za

  --- Transport Adapters for phys-schost-2 ---

  Transport Adapter:                           hme1
```

EXAMPLE 1 Displaying Cluster Configuration Information *(Continued)*

```
    Adapter State:                              Enabled
    Adapter Transport Type:                     dlpi
    Adapter Property(device_name):              hme
    Adapter Property(device_instance):          1
    Adapter Property(lazy_free):                0
    Adapter Property(dlpi_heartbeat_timeout):   10000
    Adapter Property(dlpi_heartbeat_quantum):   1000
    Adapter Property(nw_bandwidth):             80
    Adapter Property(bandwidth):                10
    Adapter Property(ip_address):               172.16.0.130
    Adapter Property(netmask):                  255.255.255.128
    Adapter Port Names:                         0
    Adapter Port State(0):                      Enabled

  Transport Adapter:                          qfe3
    Adapter State:                              Enabled
    Adapter Transport Type:                     dlpi
    Adapter Property(device_name):              qfe
    Adapter Property(device_instance):          3
    Adapter Property(lazy_free):                1
    Adapter Property(dlpi_heartbeat_timeout):   10000
    Adapter Property(dlpi_heartbeat_quantum):   1000
    Adapter Property(nw_bandwidth):             80
    Adapter Property(bandwidth):                10
    Adapter Property(ip_address):               172.16.1.2
    Adapter Property(netmask):                  255.255.255.128
    Adapter Port Names:                         0
    Adapter Port State(0):                      Enabled

  --- SNMP MIB Configuration on phys-schost-2 ---

  SNMP MIB Name:                              Event
    State:                                      SNMPv

  --- SNMP Host Configuration on phys-schost-2 ---

  --- SNMP User Configuration on phys-schost-2 ---
=== Transport Cables ===

Transport Cable:                              phys-schost-1:hme1,switch1@1
  Cable Endpoint1:                            phys-schost-1:hme1
  Cable Endpoint2:                            switch1@1
  Cable State:                                Enabled

Transport Cable:                              phys-schost-1:qfe3,switch2@1
  Cable Endpoint1:                            phys-schost-1:qfe3
```

EXAMPLE 1 Displaying Cluster Configuration Information *(Continued)*

```
  Cable Endpoint2:                              switch2@1
  Cable State:                                  Enabled

Transport Cable:                              phys-schost-2:hme1,switch1@2
  Cable Endpoint1:                              phys-schost-2:hme1
  Cable Endpoint2:                              switch1@2
  Cable State:                                  Enabled

Transport Cable:                              phys-schost-2:qfe3,switch2@2
  Cable Endpoint1:                              phys-schost-2:qfe3
  Cable Endpoint2:                              switch2@2
  Cable State:                                  Enabled

=== Transport Switches ===

Transport Switch:                             switch1
  Switch State:                                 Enabled
  Switch Type:                                  switch
  Switch Port Names:                            1 2
  Switch Port State(1):                         Enabled
  Switch Port State(2):                         Enabled

Transport Switch:                             switch2
  Switch State:                                 Enabled
  Switch Type:                                  switch
  Switch Port Names:                            1 2
  Switch Port State(1):                         Enabled
  Switch Port State(2):                         Enabled

=== Quorum Devices ===

Quorum Device Name:                           d3
  Enabled:                                      yes
  Votes:                                        1
  Global Name:                                  /dev/did/rdsk/d3s2
  Type:                                         scsi
  Access Mode:                                  scsi2
  Hosts (enabled):                              phys-schost-1, phys-schost-2

=== Device Groups ===

Device Group Name:                            db1
  Type:                                         SVM
  failback:                                     no
  Node List:                                    phys-schost-1, phys-schost-2
  preferenced:                                  yes
```

EXAMPLE 1 Displaying Cluster Configuration Information *(Continued)*

```
  numsecondaries:                                 1
  diskset name:                                   db1

=== Registered Resource Types ===

Resource Type:                                  SUNW.LogicalHostname:2
  RT_description:                                 Logical Hostname Resource Type
  RT_version:                                     2
  API_version:                                    2
  RT_basedir:                                     /usr/cluster/lib/rgm/rt/hafoip
  Single_instance:                                False
  Proxy:                                          False
  Init_nodes:                                     All potential masters
  Installed_nodes:                                <All>
  Failover:                                       True
  Pkglist:                                        SUNWscu
  RT_system:                                      True

Resource Type:                                  SUNW.SharedAddress:2
  RT_description:                                 HA Shared Address Resource Type
  RT_version:                                     2
  API_version:                                    2
  RT_basedir:                                     /usr/cluster/lib/rgm/rt/hascip
  Single_instance:                                False
  Proxy:                                          False
  Init_nodes:                                     <Unknown>
  Installed_nodes:                                <All>
  Failover:                                       True
  Pkglist:                                        SUNWscu
  RT_system:                                      True

Resource Type:                                  SUNW.qfs
  RT_description:                                 SAM-QFS Agent on SunCluster
  RT_version:                                     3.1
  API_version:                                    3
  RT_basedir:                                     /opt/SUNWsamfs/sc/bin
  Single_instance:                                False
  Proxy:                                          False
  Init_nodes:                                     All potential masters
  Installed_nodes:                                <All>
  Failover:                                       True
  Pkglist:                                        <NULL>
  RT_system:                                      False

=== Resource Groups and Resources ===
```

EXAMPLE 1 Displaying Cluster Configuration Information *(Continued)*

```
Resource Group:                                 qfs-rg
  RG_description:                                  <NULL>
  RG_mode:                                         Failover
  RG_state:                                        Managed
  Failback:                                        False
  Nodelist:                                        phys-schost-2 phys-schost-1

  --- Resources for Group qfs-rg ---

  Resource:                                     qfs-res
    Type:                                          SUNW.qfs
    Type_version:                                  3.1
    Group:                                         qfs-rg
    R_description:
    Resource_project_name:                         default
    Enabled{phys-schost-2}:                        True
    Enabled{phys-schost-1}:                        True
    Monitored{phys-schost-2}:                      True
    Monitored{phys-schost-1}:                      True

=== DID Device Instances ===

DID Device Name:                /dev/did/rdsk/d1
  Full Device Path:                phys-schost-1:/dev/rdsk/c0t0d0
  Replication:                     none
  default_fencing:                 global

DID Device Name:                /dev/did/rdsk/d2
  Full Device Path:                phys-schost-1:/dev/rdsk/c0t6d0
  Replication:                     none
  default_fencing:                 global

DID Device Name:                /dev/did/rdsk/d3
  Full Device Path:                phys-schost-2:/dev/rdsk/c1t1d0
  Full Device Path:                phys-schost-1:/dev/rdsk/c1t1d0
  Replication:                     none
  default_fencing:                 global

DID Device Name:                /dev/did/rdsk/d4
  Full Device Path:                phys-schost-2:/dev/rdsk/c1t2d0
  Full Device Path:                phys-schost-1:/dev/rdsk/c1t2d0
  Replication:                     none
  default_fencing:                 global

DID Device Name:                /dev/did/rdsk/d5
  Full Device Path:                phys-schost-2:/dev/rdsk/c1t3d0
```

EXAMPLE 1 Displaying Cluster Configuration Information *(Continued)*

```
  Full Device Path:                phys-schost-1:/dev/rdsk/c1t3d0
  Replication:                     none
  default_fencing:                 global

DID Device Name:                 /dev/did/rdsk/d6
  Full Device Path:                phys-schost-2:/dev/rdsk/
                                   c6t60020F2000004B843BC38D21000A3116d0
  Full Device Path:                phys-schost-1:/dev/rdsk/
                                   c6t60020F2000004B843BC38D21000A3116d0
  Replication:                     none
  default_fencing:                 scsi3

DID Device Name:                 /dev/did/rdsk/d7
  Full Device Path:                phys-schost-2:/dev/rdsk/
                                   c6t60020F2000004B843BC3746B000BB4A0d0
  Full Device Path:                phys-schost-1:/dev/rdsk/
                                   c6t60020F2000004B843BC3746B000BB4A0d0
  Replication:                     none
  default_fencing:                 global

DID Device Name:                 /dev/did/rdsk/d8
  Full Device Path:                phys-schost-2:/dev/rdsk/
                                   c6t60020F2000004B843BC37F8600083E05d0
  Full Device Path:                phys-schost-1:/dev/rdsk/
                                   c6t60020F2000004B843BC37F8600083E05d0
  Replication:                     none
  default_fencing:                 global

DID Device Name:                 /dev/did/rdsk/d9
  Full Device Path:                phys-schost-2:/dev/rdsk/
                                   c6t60020F2000004B843BC373F10005A987d0
  Full Device Path:                phys-schost-1:/dev/rdsk/
                                   c6t60020F2000004B843BC373F10005A987d0
  Replication:                     none
  default_fencing:                 global

DID Device Name:                 /dev/did/rdsk/d10
  Full Device Path:                phys-schost-2:/dev/rdsk/c3t50020F2300004677d1
  Full Device Path:                phys-schost-1:/dev/rdsk/c3t50020F2300004677d1
  Replication:                     none
  default_fencing:                 global

DID Device Name:                 /dev/did/rdsk/d11
  Full Device Path:                phys-schost-2:/dev/rdsk/c3t50020F2300004677d0
  Full Device Path:                phys-schost-1:/dev/rdsk/c3t50020F2300004677d0
  Replication:                     none
```

EXAMPLE 1 Displaying Cluster Configuration Information *(Continued)*

```
    default_fencing:                global

  DID Device Name:                /dev/did/rdsk/d12
    Full Device Path:                phys-schost-2:/dev/rdsk/c0t0d0
    Replication:                     none
    default_fencing:                 global

  DID Device Name:                /dev/did/rdsk/d13
    Full Device Path:                phys-schost-2:/dev/rdsk/c0t1d0
    Replication:                     none
    default_fencing:                 global

  DID Device Name:                /dev/did/rdsk/d14
    Full Device Path:                phys-schost-2:/dev/rdsk/c0t6d0
    Replication:                     none
    default_fencing:                 global

  === NAS Devices ===

  === Telemetry Attributes ===
```

EXAMPLE 2 Displaying Configuration Information About Selected Cluster Components

The following command displays information about resources, resource types, and resource groups. Information is displayed for only the cluster.

```
# cluster show -t resource,resourcetype,resourcegroup
  Single_instance:                              False
  Proxy:                                        False
  Init_nodes:                                   <Unknown>
  Installed_nodes:                              <All>
  Failover:                                     True
  Pkglist:                                      SUNWscu
  RT_system:                                    True

Resource Type:                                 SUNW.qfs
  RT_description:                               SAM-QFS Agent on SunCluster
  RT_version:                                   3.1
  API_version:                                  3
  RT_basedir:                                   /opt/SUNWsamfs/sc/bin
  Single_instance:                              False
  Proxy:                                        False
  Init_nodes:                                   All potential masters
  Installed_nodes:                              <All>
  Failover:                                     True
  Pkglist:                                      <NULL>
```

EXAMPLE 2 Displaying Configuration Information About Selected Cluster Components *(Continued)*

```
  RT_system:                                      False

=== Resource Groups and Resources ===

Resource Group:                                 qfs-rg
  RG_description:                                 <NULL>
  RG_mode:                                        Failover
  RG_state:                                       Managed
  Failback:                                       False
  Nodelist:                                       phys-schost-2 phys-schost-1

  --- Resources for Group qfs-rg ---

  Resource:                                     qfs-res
    Type:                                         SUNW.qfs
    Type_version:                                 3.1
    Group:                                        qfs-rg
    R_description:
    Resource_project_name:                        default
    Enabled{phys-schost-2}:                       True
    Enabled{phys-schost-1}:                       True
    Monitored{phys-schost-2}:                     True
    Monitored{phys-schost-1}:                     True
```

EXAMPLE 3 Displaying Cluster Status

The following command displays the status of all cluster nodes.

```
# cluster status -t node
=== Cluster Nodes ===

--- Node Status ---

Node Name                                       Status
---------                                       ------
phys-schost-1                                   Online
phys-schost-2                                   Online

--- Node Status ---

Node Name                                       Status
---------                                       ------
```

Alternately, you can also display the same information by using the `clnode` command.

```
# clnode status
=== Cluster Nodes ===
```

EXAMPLE 3 Displaying Cluster Status *(Continued)*

```
--- Node Status ---

Node Name                                       Status
---------                                       ------
phys-schost-1                                   Online
phys-schost-2                                   Online
```

EXAMPLE 4 Creating a Cluster

The following command creates a cluster that is named `cluster-1` from the cluster configuration file `suncluster.xml`.

```
# cluster create -i /suncluster.xml cluster-1
```

EXAMPLE 5 Changing a Cluster Name

The following command changes the name of the cluster to `cluster-2`.

```
# cluster rename -c cluster-2
```

EXAMPLE 6 Disabling a Cluster's `installmode` Property

The following command disables a cluster's `installmode` property.

```
# cluster set -p installmode=disabled
```

EXAMPLE 7 Modifying the Private Network

The following command modifies the private network settings of a cluster. The command sets the private network address to `172.10.0.0`. The command also calculates and sets a minimum private netmask to support the specified eight nodes and four private networks.

```
# cluster set-netprops \
      -p private_netaddr=172.10.0.0 -p max_nodes=8 \
      -p max_privatenets=4
```

You can also specify this command as follows:

```
# cluster set-netprops \
      -p private_netaddr=172.10.0.0,max_nodes=8,\
       max_privatenets=4
```

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Interface Stability	Evolving

See Also Intro(1CL), `init`(1M), `su`(1M), scha_calls(3HA), `attributes`(5), `rbac`(5), clconfiguration(5CL)

Notes The superuser can run all forms of this command.

All users can run this command with the `-?` (help) or `-V` (version) option.

To run the `cluster` command with subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`create`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`list-cmds`	`solaris.cluster.read`
`rename`	`solaris.cluster.modify`
`restore-netprops`	`solaris.cluster.modify`
`set`	`solaris.cluster.modify`
`set-netprops`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`
`show-netprops`	`solaris.cluster.read`
`shutdown`	`solaris.cluster.admin`
`status`	`solaris.cluster.read`

Name clvxvm – configure VERITAS Volume Manager for Sun Cluster

Synopsis

```
/usr/cluster/bin/clvxvm  -V

/usr/cluster/bin/clvxvm  [subcommand] -?

/usr/cluster/bin/clvxvm  subcommand -v

/usr/cluster/bin/clvxvm encapsulate

/usr/cluster/bin/clvxvm initialize
```

Description The `clvxvm` utility initializes VERITAS Volume Manager (VxVM) on a Sun Cluster node and optionally performs root-disk encapsulation. There is no short form of this command name.

You can only use this utility with VxVM versions 4.1 or later. For older versions of VxVM, instead use the scvxinstall(1M) utility.

The general form of this command is as follows:

`clvxvm` [*subcommand*] [*options*]

You can omit *subcommand* only if *options* specifies the `-?` option or the `-V` option.

Each option of this command has a long form and a short form. Both forms of each option are given with the description of the option in the OPTIONS section of this man page.

You can only use this command from a node that is booted in cluster mode. All nodes in the cluster configuration must be current cluster members. All nodes must be added to the node authentication list.

You can use this command only in the global zone.

Subcommands The following subcommands are supported:

`encapsulate`
Encapsulates the root disk and performs other Sun Cluster-specific tasks.

The `encapsulate` subcommand performs the following tasks:

- Verifies that the current node is booted in cluster mode and that all other cluster nodes are running in cluster mode.
- Verifies that the user is superuser.
- Enforces a cluster-wide value for the `vxio` major number by modifying the node's `/etc/name_to_major` file, if necessary. This task ensures that the `vxio` number is the same on all cluster nodes.
- Runs several VxVM commands to prepare for root-disk encapsulation.

- Modifies the global-devices entry in the `/etc/vfstab` file that is specified for the `/global/.devices/node@`*n* file system, where *n* is the node ID number. The `clvxvm` utility replaces the existing device path `/dev/did/{r}dsk` with `/dev/{r}dsk`. This change ensures that VxVM recognizes that the global-devices file system resides on the root disk.
- Twice reboots each node that is running `clvxvm`. The first reboot allows VxVM to complete the encapsulation process. The second reboot resumes normal operation. The `clvxvm` utility includes a synchronization mechanism to ensure that the utility reboots only one node at a time, to prevent loss of quorum.
- Unmounts the global-devices file system. The file system is automatically remounted after the encapsulation process is complete.
- Recreates the special files for the root-disk volumes with a unique minor number on each node.

This subcommand expects that VxVM packages and licenses are already installed and that the VxVM configuration daemon is successfully enabled on this node.

Each root disk that you encapsulate must have at least two free (unassigned) partitions.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

`initialize`
Initializes VxVM and performs other Sun Cluster-specific tasks.

The `initialize` subcommand performs the following tasks:

- Verifies that the current node is booted in cluster mode and that all other cluster nodes are running in cluster mode.
- Verifies that the user is superuser.
- Enforces a cluster-wide value for the `vxio` major number by modifying the node's `/etc/name_to_major` file, if necessary. This task ensures that the `vxio` number is the same on all cluster nodes.
- Instructs the user to reboot the node to resume operation with the new `vxio` major number in effect, in case the number had to be changed.

This subcommand expects that VxVM packages and licenses are already installed and that the VxVM configuration daemon is successfully enabled on this node.

Users other than superuser require `solaris.cluster.modify` RBAC authorization to use this subcommand.

Options The following options are supported:

`-?`
`---help`
Displays help information.

You can use this option either alone or with a subcommand.

- If you use this option alone, the list of available subcommands is printed.
- If you use this option with a subcommand, the usage options for that subcommand are printed.

When you use this option, no other processing is performed.

`-V`
`---version`
Displays the version of the command.

Do not specify this option with subcommands, operands, or other options. The subcommands, operands, or other options are ignored. The `-V` option only displays the version of the command. No other operations are performed.

`-v`
`---verbose`
Displays verbose information to standard output.

You can use this option with any form of the command.

Operands No form of this command accepts an operand.

Exit Status The complete set of exit status codes for all commands in this command set are listed on the Intro(1CL) man page.

This command returns the following exit status codes:

`0 CL_NOERR`
No error

`1 CL_ENOMEM`
Not enough swap space

`3 CL_EINVAL`
Invalid argument

`6 CL_EACCESS`
Permission denied

`35 CL_EIO`
I/O error

Examples **EXAMPLE 1** Initializing VxVM on a Cluster Node

The following example shows how to initialize VxVM the cluster node from which the command is issued.

```
# clvxvm initialize
```

EXAMPLE 2 Initializing VxVM on a Cluster Node and Encapsulating the Root Disk

The following example shows how to initialize VxVM and encapsulate the root disk on the cluster node from which the command is issued.

```
# clvxvm encapsulate
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

Files

`/var/cluster/cmvxvm/*`
Location of temporary files that are used by the `clvxvm` utility.

`/var/cluster/logs/install/clvxvm.log.`*pid*
Log file that is created by the `scvxinstall` utility.

See Also

Intro(1CL), cldevice(1CL), cldevicegroup(1CL), clsetup(1CL), cluster(1CL), scinstall(1M), scvxinstall(1M), `rbac`(5)

Sun Cluster Software Installation Guide for Solaris OS

Notes

The superuser can run any forms of this command.

Any user can also run this command with the following options:

- `-?` (help) option
- `-V` (version) option

To run this command with other subcommands, users other than superuser require RBAC authorizations. See the following table.

Subcommand	RBAC Authorization
`encapsulate`	`solaris.cluster.modify`
`initialize`	`solaris.cluster.modify`

REFERENCE

SC32 1ha

Name rt_callbacks – callback interface for management of services as Sun Cluster resources

Synopsis

```
method-path -R resource -T type -G group [-Z zonename]

validate-path [-c | -u] -R resource -T type -G group
[-r prop=val] [-x prop=val] [-g prop=val] [-Z zonename]
```

Description The callback interface for Sun Cluster resource types defines the interface that the Resource Group Manager (RGM) uses to control services as cluster resources. The implementor of a resource type provides programs or scripts that serve as the callback methods.

method-path The path to the program that is declared in the resource-type registration file. This program is registered with the `clresourcetype` command as one of the following callback methods for the resource type: `START`, `STOP`, `INIT`, `FINI`, `BOOT`, `PRENET_START`, `POSTNET_STOP`, `MONITOR_START`, `MONITOR_STOP`, `MONITOR_CHECK`, or `UPDATE`. See the clresourcetype(1CL) and rt_reg(4) man pages.

validate-path The path to the program that is declared as a resource type's `VALIDATE` method in the resource-type registration file. This program is registered with the `clresourcetype` command.

The callback methods are passed prescribed options and are expected to take specific actions to control the operation of the service on the cluster.

The resource-type developer declares the paths to the callback method programs in the resource-type registration file. The cluster administrator uses the `clresourcetype` command to register the resource type in the cluster configuration. The cluster administrator can then use this registered resource type to create resources. These resources are configured in resource groups that the RGM manages.

The RGM responds to events by automatically invoking the callback methods of the resources in the resource groups that the RGM manages. The callback methods are expected to take specific actions on the service that is represented by the resource. Examples of these actions include stopping and starting the service on a cluster node or zone.

The exit status code that is returned from the callback method indicates to the RGM whether the callback method succeeded or failed. The RGM takes action if a method fails, or it reports the failure in the resource state. As a result, the cluster administrator can note the failure and take appropriate action.

Options The following options are supported:

`-c`

Specifies that the method is called when the cluster administrator creates the resource to validate the initial settings of *all* resource and resource-group properties.

The RGM specifies either the `-c` option or the `-u` option, but never both options at the same time.

When the cluster administrator creates a resource and the `VALIDATE` method is called, *all* system-defined, extension, and resource-group properties are passed to the `VALIDATE` method. When the cluster administrator updates a resource and the `VALIDATE` method is called, only the properties that are being updated are passed to the `VALIDATE` method.

`-g` *prop=val*
Specifies the value of a resource-group property that is passed to a `VALIDATE` method.

prop The name of a resource-group property.

val The value that is passed to the method when the cluster administrator creates or updates the resource.

`-G` *group*
Specifies the name of the resource group in which the resource is configured.

`-r` *prop=val*
Specifies the value of a system-defined resource property that is passed to a `VALIDATE` method.

prop The name of a system-defined resource property.

val The value that is passed to the method when the cluster administrator creates or updates the resource.

`-R` *resource*
Specifies the name of the resource for which the method is invoked.

`-T` *type*
Specifies the name of the resource type of the resource.

`-u`
Specifies that the method is called when the cluster administrator updates a property of an existing resource or an existing resource group.

The RGM specifies either the `-c` option or the `-u` option, but never both options at the same time.

When the cluster administrator creates a resource and the `VALIDATE` method is called, *all* system-defined, extension, and resource-group properties are passed to the `VALIDATE` method. When the cluster administrator updates a resource and the `VALIDATE` method is called, only the properties that are being updated are passed to the `VALIDATE` method.

`-x` *prop=val*
Specifies the value of a resource extension property that is passed to a `VALIDATE` method.

prop The name of a resource extension property. An extension property is defined by the resource-type implementation. This extension property is declared in the parameters table of the resource-type registration file.

val The value that is passed to the method when the cluster administrator creates or updates the resource.

`-Z` *zonename*
Specifies the name of the non-global zone in which a resource group is configured to run.

If the `Global_zone` resource-type property is set to `TRUE`, methods execute in the global zone, even if the resource group that contains the resource runs in a non-global zone. This option provides the name of the non-global zone in which the resource group is configured to run.

The `-Z` option is not passed whenever either of the following conditions is met:

- The `Global_zone` property is set to `FALSE`.
- The resource group is configured to run in the global zone.

Usage The callback methods are defined by the RGM mechanism that invokes them. These methods are expected to execute operations on a cluster resource. These methods are also expected to return an exit status that reports whether the method succeeded or failed. The following section describes each callback method.

`BOOT`
This method is invoked when a node or zone joins or rejoins the cluster when it is booted or rebooted. This method is called on nodes or zones that are specified by the `Init_nodes` resource-type property. Similar to `INIT`, this method is intended to initialize the resource on nodes or zones that join the cluster after the resource group that contains the resource has been brought online. This method is invoked on resources that are in managed resource groups but not on resources that are in unmanaged resource groups.

`FINI`
This method is invoked when the resource group that contains the resource is removed from management by the RGM. This method is called on nodes or zones that are specified by the `Init_nodes` resource-type property. This method unconfigures the resource and cleans up any persistent settings that are made by the `INIT` method or the `BOOT` method.

`INIT`
This method is invoked when the resource group that contains the resource is put under the management of the RGM. This method is called on nodes or zones that are specified by the `Init_nodes` resource-type property. This method initializes the resource.

`MONITOR_CHECK`
This method is called before the resource group that contains the resource is relocated to a new node or zone. This method is called when the fault monitor executes the `GIVEOVER` option of either the `scha_control` command or the `scha_control()` function. See the scha_control(1HA) and the scha_control(3HA) man pages.

This method is called on any node or zone that is a potential new master for the resource group. The MONITOR_CHECK method assesses whether a node or zone is healthy enough to run a resource. The MONITOR_CHECK method must be implemented in such a way that it does not conflict with the running of another method concurrently.

If the MONITOR_CHECK method fails, it vetoes the relocation of the resource group to the node or zone where the callback was invoked.

MONITOR_START
: This method is called after the resource is started, on the same node or zone where the resource is started. This method starts a monitor for the resource.

  MONITOR_START failure causes the RGM to set the resource state to MONITOR_FAILED.

MONITOR_STOP
: This method is called before the resource is stopped, on the same node or zone where the resource is running. This method stops a monitor for the resource. This method is also called when monitoring is disabled by the cluster administrator.

  The action that the RGM takes when a MONITOR_STOP method fails depends on the setting of the Failover_mode property for the resource. If Failover_mode is set to HARD, the RGM attempts to forcibly stop the resource by rebooting the node or zone. Otherwise, the RGM sets the resource's state to STOP_FAILED.

POSTNET_STOP
: An auxiliary to the STOP method, this method is intended to perform shutdown actions that are needed after the related network address is configured down. This method is called on nodes or zones where the STOP method has been called. This method is invoked after the network addresses in the resource group have been configured down, and after the STOP method for the resource has been called. However, this method is invoked before the network addresses have been unplumbed. The POSTNET_STOP method is called after the STOP method for the resource and after the POSTNET_STOP method of any resource that depends on the resource.

  The action that the RGM takes when a POSTNET_STOP method fails depends on the setting of the Failover_mode property for the resource. If Failover_mode is set to HARD, the RGM attempts to forcibly stop the resource by aborting the node or zone. Otherwise, the RGM sets the resource's state to STOP_FAILED.

PRENET_START
: An auxiliary to the START method, this method is intended to perform startup actions that are needed before the related network address is configured up. This method is called on nodes or zones where the START method is to be called. This method is invoked after network addresses in the same resource group have been plumbed. However, this method is invoked before the addresses have been configured up and before the START method for the resource is called. The PRENET_START method is called before the START method for the resource and before the PRENET_START method of any resource that depends on the resource.

The action that the RGM takes when a PRENET_START method fails depends on the setting of the Failover_mode property for the resource. If Failover_mode is set to SOFT or HARD, the RGM attempts to relocate the resource group that contains the resource to another node or zone. Otherwise, the RGM sets the resource's state to START_FAILED.

START
: This method is invoked on a cluster node or zone when the resource group that contains the resource is brought online on that node or zone. The cluster administrator can toggle the state between on and off by using the clresourcegroup command. The START method activates the resource on a node or zone.

  The action that the RGM takes when a START method fails depends on the setting of the Failover_mode property for the resource. If Failover_mode is set to SOFT or HARD, the RGM attempts to relocate the resource's group to another node or zone. Otherwise, the RGM sets the resource's state to START_FAILED.

STOP
: This method is invoked on a cluster node or zone when the resource group that contains the resource is brought offline on that node or zone. The cluster administrator can toggle the state between on and off by using the clresourcegroup command. This method deactivates the resource if the resource is active.

  The action that the RGM takes when a STOP method fails depends on the setting of the Failover_mode property for the resource. If Failover_mode is set to HARD, the RGM attempts to forcibly stop the resource by rebooting the node or zone. Otherwise, the RGM sets the resource's state to STOP_FAILED.

UPDATE
: This method is called to notify a running resource that properties have been changed. The UPDATE method is invoked after the RGM succeeds in setting properties of a resource or its resource group. This method is called on nodes or zones where the resource is online. This method can call the scha_resource_get and scha_resourcegroup_get commands to read property values that can affect an active resource and adjust the running resource accordingly.

VALIDATE
: This method is called when a resource is created or when a resource or its containing resource group is updated. VALIDATE is called on the set of cluster nodes or zones that are specified by the Init_nodes property of the resource's type.

  The VALIDATE method is called before the creation or update of the resource is applied. If the method fails on a node or zone and a failure exit status code is generated, the creation or update is canceled.

  When the cluster administrator creates a resource and the VALIDATE method is called, *all* system-defined, extension, and resource-group properties are passed to the VALIDATE method. When the cluster administrator updates a resource and the VALIDATE method is

called, only the properties that are being updated are passed to the `VALIDATE` method. You can use the `scha_resource_get` and `scha_resourcegroup_get` commands to retrieve the properties of the resource that are not being updated.

When you implement the `VALIDATE` method, any message that you write to `stdout` or `stderr` is passed back to the user command. This action is useful to explain the reason for a validation failure or to provide instructions to the user regarding the resource.

Environment Variables The Sun Cluster resource management callback methods are executed with superuser permission by the RGM. The programs that implement the methods are expected to be installed with appropriate execution permissions, and for security, should not be writable.

Environment variables that are set for callback method execution are as follows:

```
HOME=/
PATH=/usr/bin:/usr/cluster/bin
LD_LIBRARY_PATH=/usr/cluster/lib
```

Signals If a callback method invocation exceeds its timeout period, the process is sent a `SIGTERM` signal. If the `SIGTERM` signal fails to stop the method execution, the process is sent a `SIGKILL` signal.

Exit Status The following exit status codes are returned.

`0` The command completed successfully.

nonzero An error occurred.

The specific value of a failure exit status does not affect the RGM's action on failure. However, the exit status is recorded in the cluster log when the method fails. A resource-type implementation might define different nonzero exit codes to communicate error information to the cluster administrator through the cluster log.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also clresourcegroup(1CL), clresourcetype(1CL), scha_cmds(1HA), scha_control(1HA), scha_resource_get(1HA), scha_resourcegroup_get(1HA), `signal`(3C), `stdio`(3C), scha_calls(3HA), scha_control(3HA), rt_reg(4), `attributes`(5)

Name scdsbuilder – launch GUI version of Sun Cluster Agent Builder

Synopsis

```
scdsbuilder
```

Description The `scdsbuilder` command launches the GUI version of the Sun Cluster Agent Builder.

Before you use Agent Builder, verify the following requirements:

- The Java runtime environment is included in your `$PATH` variable. Agent Builder depends on the Java Development Kit, starting with Version 1.3.1. If the Java Development Kit is not included in your `$PATH` variable, the Agent Builder command (`scdsbuilder`) returns and displays an error message.
- You have installed the Developer System Support software group of the Solaris 9 OS or Solaris 10 OS.
- The `cc` compiler is included in your `$PATH` variable. Agent Builder uses the first occurrence of `cc` in your `$PATH` variable to identify the compiler with which to generate C binary code for the resource type. If `cc` is not included in `$PATH`, Agent Builder disables the option to generate C code.

Note – You can use a different compiler with Agent Builder than the standard `cc` compiler. To use a different compiler, create a symbolic link in `$PATH` from `cc` to a different compiler, such as `gcc`. Or, change the compiler specification in the makefile (currently, `CC=cc`) to the complete path for a different compiler. For example, in the makefile that is generated by Agent Builder, change `CC=cc` to `CC=`*pathname*`/gcc`. In this case, you cannot run Agent Builder directly. Instead, you must use the `make` and `make pkg` commands to generate data service code and the package.

Exit Status This command returns the following exit status codes:

`0`	The command completed succesfully.
nonzero	An error occurred.

Files

install_directory`/rtconfig`	Contains information from the previous session. This information facilitates the tool's quit and restart feature.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also `cc`(1B), scdscreate(1HA), scdsconfig(1HA), `attributes`(5)

Chapter 9, "Sun Cluster Agent Builder," in *Sun Cluster Data Services Developer's Guide for Solaris OS*

Name scdsconfig – configure resource type template

Synopsis
```
scdsconfig  -s start_command [-u start_method_timeout]
 [-e validate_command] [-y validate_method_timeout]
 [-t stop_command] [-v stop_method_timeout] [-m probe_command]
 [-n probe_timeout] [-d working_directory]
```

Description The `scdsconfig` command configures the resource type template that you created with the scdscreate(1HA) command. The `scdsconfig` command enables you to configure C, Generic Data Service (GDS), or Korn shell-based templates for both network aware (client-server model) and non-network aware (clientless) applications.

The `scdsconfig` command configures application-specific commands to start, stop, validate, and probe the application. You can also use the `scdsconfig` command to set timeout values for the `start`, `stop`, `validate`, and `probe` commands. The `scdsconfig` command supports both network aware (client-server model) and non-network aware (clientless) applications. You can run the `scdsconfig` command from the same directory where the `scdscreate` command was run. You can also specify that same directory by using the `-d` option. The `scdsconfig` command configures the resource type template by placing the user-specified parameters at correct locations in the generated code. If C was the type of generated source code, this command also compiles the code. The `scdsconfig` command puts the output into a Solaris package that you can then install. This command creates the package in the `pkg` subdirectory under the $*vendor_id*$*resource_type_name* directory created by the `scdscreate` command.

Options The following options are supported:

`-d` *working_directory*
: Specifies the directory where the `scdscreate` command was run.

  You must specify this option if you run the `scdsconfig` command from a directory other than the directory where the `scdscreate` command was run.

`-e` *validate_command*
: Specifies the absolute path to a command to invoke to validate the application. If you do not specify an absolute path, the application is not validated. The *validate_command* returns with an exit status of `0` if the application is running successfully. An exit status other than `0` indicates that the application is failing to perform correctly. In this case, one of two results occur, depending on the failure history of the application in the past:

  - The resources of this resource type are either restarted on the same node or zone.
  - The resource group that contains the resource has failed over to another healthy node or zone.

`-m` *probe_command*	Specifies a command to periodically check the health of the network aware or non-network aware application. It must be a complete command line that can be passed directly to a shell to probe the application. The *probe_command* returns with an exit status of `0` if the application is running successfully. An exit status other than `0` indicates that the application is failing to perform correctly. In this case, one of two results occur, depending on the failure history of the application in the past: ■ The resources of this resource type are either restarted on the same node or zone. ■ The resource group that contains the resource is failed over to another healthy node or zone.
`-n` *probe_timeout*	Specifies the timeout, in seconds, for the probe command. The timeout must take into account system overloads to prevent false failures. The default value is 30 seconds.
`-s` *start_command*	Specifies the command that starts the application. The start command must be a complete command line that can be passed directly to a shell to start the application. You can include command-line arguments to specify host names, port numbers, or other configuration data that is necessary to start the application. To create a resource type with multiple independent process trees, you specify a text file that contains the list of commands, one per line, to start the different process trees.
`-t` *stop_command*	Specifies the stop command for the application. The stop command must be a complete command line that can be passed directly to a shell to stop the application. If you omit this option, the generated code stops the application by issuing signals. The stop command is allotted 80 percent of the timeout value to stop the application. If the stop command fails to stop the application within this period, a `SIGKILL` is allotted 15 percent of the timeout value to stop the application. If `SIGKILL` also fails to stop the application, the stop method returns with an error.
`-u` *start_method_timeout*	Specifies the timeout, in seconds, for the start command. The timeout must take into account system overloads to prevent false failures. The default value is 300 seconds.

`-v` *stop_method_timeout* — Specifies the timeout, in seconds, for the stop command. The timeout must take into account system overloads to prevent false failures. The default value is 300 seconds.

`-y` *validate_method_timeout* — Specifies the timeout, in seconds, for the validate command. The timeout must take into account system overloads to prevent false failures. The default value is 300 seconds.

Exit Status

The following exit status codes are returned:

`0` — The command completed successfully.

nonzero — An error occurred.

Files

working_directory`/rtconfig` — Contains information from the previous session. Facilitates the tool's quit and restart feature.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Evolving

See Also

`ksh`(1), scdsbuilder(1HA), scdscreate(1HA), `attributes`(5)

Name scdscreate – create a Sun Cluster resource type template

Synopsis
```
scdscreate  -V vendor_id -T resource_type_name -a [-s]
 [-n RT_version] [-d working_directory] [-k |  -g]
```

Description The `scdscreate` command creates a template for making an application highly available (HA) or scalable. This command enables you to create C-, Generic Data Service (GDS)-, or Korn shell-based templates for both network aware (client-server model) and non-network aware (clientless) applications.

You can create the template in one of two fundamentally different ways:

GDS	`scdscreate` creates a set of three driving scripts that work from a single resource type `SUNW.gds`, which is pre-installed on the cluster. These scripts are named start*RT_Name*, stop*RT_Name*, and remove*RT_Name* and starts, stops, and removes an instance of that application. In this model, the implementation of the `SUNW.gds` resource type that is pre-installed on the cluster is immutable.
Generated Source Code	`scdscreate` creates a template for a Sun Cluster resource type, whose instantiations run under the control of the Resource Group Manager (RGM) to make the given application highly available and scalable.

Either model can create templates for network aware (client-server model) and non-network aware (client-less) applications.

`scdscreate` creates a directory of the form $*vendor_id*$*resource_type_name* under *working_directory*. This directory contains the driving scripts, or the generated source, binary, and package files for the resource type. `scdscreate` also creates a configuration file, `rtconfig`, in which you can store configuration information for the resource type. `scdscreate` allows you to create only one resource type per directory. You must create different resource types in different directories.

Options The following options are supported:

`-a`	This parameter specifies that the resource type that is being created is not network aware. `scdscreate` disables all the networking related code in the template that is created.
`-n` *RT_version*	This optional parameter specifies the version of the generated resource's type. If you omit this parameter, and you're creating a C- or Korn shell-based application, the text string `1.0` is used by default. If you omit this parameter, and you're creating a GDS-based application, the `RT_version` string of the GDS is used by default. The *RT_version*

distinguishes between multiple registered versions, or upgrades, of the same base resource type.

You cannot include the following characters in *RT_version*: blank, tab, slash (`/`), backslash (`\`), asterisk (`*`), question mark (`?`), comma (`,`), semicolon (`;`), left square bracket (`[`), or right square bracket (`]`).

`-d` *working_directory* — Creates the template for the resource type in a directory other than the current directory. If you omit this argument, `scdscreate` creates the template in the current directory.

`-g` — This optional parameter generates the GDS-based form of the template to make an application highly available or scalable.

`-k` — This optional parameter generates source code in Korn shell command syntax rather than in C. See `ksh`(1).

`-s` — This optional parameter indicates that the resource type is scalable. You can configure an instance (resource) of a scalable resource type into a failover resource group, and hence, turn off the scalability feature. If you omit this argument, `scdscreate` creates the template for a failover resource type.

`-T` *resource_type_name* — The resource type name and resource type version, in conjunction with the vendor ID, uniquely identifies the resource type that is being created.

`-V` *vendor_id* — The vendor ID is typically the stock symbol, or some other identifier of the vendor that is creating the resource type. `scdscreate` affixes the vendor ID, followed by a period (`.`) to the beginning of the resource type name. This syntax ensures that the resource type name remains unique if more than one vendor uses the same resource type name.

Exit Status

`0` — The command completed successfully.

nonzero — An error occurred.

Files

working_directory`/rtconfig` — Contains information from the previous session and facilitates the quit and restart feature of `scdscreate`.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also ksh(1), scdsbuilder(1HA), scdsconfig(1HA), attributes(5), rt_properties(5)

Sun Cluster Data Services Developer's Guide for Solaris OS

Name scha_cluster_get – access cluster information

Synopsis

```
scha_cluster_get  -O optag [args]
```

Description The `scha_cluster_get` command accesses and generates information about a cluster. You can access information about the cluster, nodes, zones, host names, resource groups, resource types, and states.

The command is intended for use in shell script implementations of callback methods for resource types. These callback methods for resource types represent services that are controlled by the cluster's Resource Group Manager (RGM). This command provides the same information as the `scha_resource_get()` function.

This command sends output to the standard output (`stdout`) in formatted strings on separate lines, as described in the scha_cmds(1HA) man page. You can store the output in shell variables. You can also parse the output with shell commands such as the `awk` command.

Options The following options are supported:

`-O` *optag*

The *optag* argument indicates the information to be accessed. Depending on the *optag*, an additional argument may be needed to indicate the cluster node or zone for which information is to be retrieved.

Note – *optag* options, such as `NODENAME_LOCAL` and `NODENAME_NODEID`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify *optag* options.

The following *optag* values are supported:

`ALL_NODEIDS`
Generates on successive lines the numeric node identifiers of all nodes in the cluster.

`ALL_NODENAMES`
Generates on successive lines the names of all nodes in the cluster.

`ALL_NONGLOBAL_ZONES`
Generates on successive lines the `nodename:zonename` string of all the non-global zones on all nodes in the cluster.

`ALL_NONGLOBAL_ZONES_NODEID`
Generates on successive lines the `nodename:zonename` string of all non-global zones on the cluster node whose numeric node identifier is given as the argument.

`ALL_PRIVATELINK_HOSTNAMES`
Generates on successive lines the host names by which all cluster nodes are addressed on the cluster interconnect.

`ALL_RESOURCEGROUPS`
Generates on successive lines the names of all the resource groups that are being managed in the cluster.

`ALL_RESOURCETYPES`
Generates on successive lines the names of all the resource types that are registered in the cluster.

`ALL_ZONES`
Generates on successive lines the `nodename:zonename` string of all zones, including the global zone, on all nodes in the cluster.

`ALL_ZONES_NODEID`
Generates on successive lines the `nodename:zonename` string of all zones, including the global zone, on the cluster node whose numeric node identifier is given as the argument.

`CLUSTERNAME`
Generates the name of the cluster.

`NODEID_LOCAL`
Generates the numeric node identifier for the node where the command is executed.

`NODEID_NODENAME`
Generates the numeric node identifier of the node indicated by the name. Requires an additional unflagged argument that is the name of a cluster node.

`NODENAME_LOCAL`
Generates the name of the cluster node where the command is executed.

`NODENAME_NODEID`
Generates the name of the cluster node indicated by the numeric identifier. Requires an additional unflagged argument that is a numeric cluster node identifier.

`NODESTATE_LOCAL`
Generates `UP` or `DOWN` depending on the state of the node where the command is executed.

`NODESTATE_NODE`
Generates `UP` or `DOWN` depending on the state of the named node. Requires an additional unflagged argument that is the name of a cluster node.

`PRIVATELINK_HOSTNAME_LOCAL`
Generates the host name by which the node on which the command is run is addressed over the cluster interconnect.

`PRIVATELINK_HOSTNAME_NODE`
: Generates the host name by which the named node is addressed on the cluster interconnect. Requires an additional unflagged argument that is the name of a cluster node.

`SYSLOG_FACILITY`
: Generates the number of the `syslog`(3C) facility that the RGM uses for log messages. The value is `24`, which corresponds to the `daemon` facility. You can use this value as the facility level in the `logger`(1) command to log messages in the cluster log.

`ZONE_LOCAL`
: Generates the name of the zone from which the command is issued. The format of the zone name that is returned is `nodename:zonename`.

Examples

EXAMPLE 1 Using the `scha_cluster` Command in a Shell Script

The following shell script uses the `scha_cluster` command to print whether each cluster node is up or down:

```
#!/bin/sh
nodenames=‘scha_cluster_get -O All_Nodenames‘
for node in $nodenames
do
     state=‘scha_cluster_get -O NodeState_Node $node‘
     printf "State of node: %s\n exit: %d\n value: %s\n" "$node" $? "$state"
done
```

Exit Status

The following exit status codes are returned:

`0` Successful completion.

nonzero An error occurred.

Error codes are described in scha_calls(3HA).

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Stable

See Also

`awk`(1), `logger`(1), `sh`(1), scha_cmds(1HA), `syslog`(3C), scha_calls(3HA), scha_cluster_get(3HA), `attributes`(5)

Name scha_cmds – command standard output for `scha_cluster_get`, `scha_control`, `scha_resource_get`, `scha_resourcegroup_get`, `scha_resourcetype_get`, `scha_resource_setstatus`

Synopsis *scha_command* `-O` *optag*...

Description The Sun Cluster scha_cluster_get(1HA), scha_control(1HA), scha_resource_get(1HA), scha_resourcegroup_get(1HA), scha_resourcetype_get(1HA), and scha_resource_setstatus(1HA) commands are command-line implementations of the callback methods for resource types. See rt_callbacks(1HA).

Resource types represent services that are controlled by the cluster's Resource Group Manager (RGM) facility. These commands provide a command-line interface to the functionality of the scha_calls(3HA) C functions.

The `get` commands access cluster configuration information. All of these commands have the same general interface. These commands all take an `-O` *optag* operand. This operand indicates the information to be accessed. These commands all send the results to the standard output (`stdout`) as formatted strings. Additional arguments might be needed depending on the command and the value of *optag*. For information about the format of different *optag* results, see the "Results Format" section.

Note – *optag* options, for all `scha` commands, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify *optag* options.

The scha_control(1HA) command also takes an `-O` *optag* option that indicates a control operation, but does not produce output to standard output.

The scha_resource_setstatus(1HA) command sets the `STATUS` and `STATUS_MSG` properties of a resource that is managed by the RGM.

Result Formats The format of strings that are output to the standard output by the commands depends on the type of the result that is indicated by the *optag* that you include with the `-O` option. Formats for each type are specified in the following table. Format notation is described in `formats`(5).

Result Type	Format on Standard Output
`boolean`	`TRUE\n` or `FALSE\n`
`enum`	`%s\n`, the string name of an `enum` value.

Result Type	Format on Standard Output
`extension`	`%s\n`, the type attribute of the extension property, which is one of the following values: `STRING`, `INT`, `BOOLEAN`, `ENUM`, or `STRINGARRAY`. Following the type information, the property value is output according to the formats for each type as follows: `STRING` as `string`, `INT` as `int`, `BOOLEAN` as `boolean`, `ENUM` as `enum`, `STRINGARRAY` as `string_array`.
`int`	`%d\n`
`status`	`%s\n%s\n`, the first string is the status, which is one of the following `enum` values: `DEGRADED`, `FAULTED`, `OFFLINE`, `ONLINE`, or `UNKNOWN`. The second string is the status message.
`string`	`%s\n`
`string_array`	Each element in the array is output in the format `%s\n`. An asterisk, indicating all nodes, zones, or resources, can be returned for the `GLOBAL_RESOURCES_USED` and `INSTALLED_NODES` properties.
`unsigned_int`	`%u\n`
`unsigned_int_array`	Each element in the array is output in the format `%u\n`

optag Result Types

The following table specifies the valid *optag* values for different commands as well as the type of the result that is output according to the formats specified in the previous table.

optag Values for scha_cluster_get(1HA)	Result Type
`ALL_NODEIDS`	`unsigned_int_array`
`ALL_NODENAMES`	`string_array`
`ALL_NONGLOBAL_ZONES`	`string`
`ALL_NONGLOBAL_ZONES_NODEID`	`string`
`ALL_PRIVATELINK_HOSTNAMES`	`string_array`
`ALL_RESOURCEGROUPS`	`string_array`
`ALL_RESOURCETYPES`	`string_array`
`ALL_ZONES`	`string`
`ALL_ZONES_NODEID`	`string`
`CLUSTERNAME`	`string`

optag Values for scha_cluster_get(1HA)	Result Type
NODEID_LOCAL	unsigned_int
NODEID_NODENAME	unsigned_int
NODENAME_LOCAL	string
NODENAME_NODEID	string
NODESTATE_LOCAL	enum (UP, DOWN)
NODESTATE_NODE	enum (UP, DOWN)
PRIVATELINK_HOSTNAME_LOCAL	string
PRIVATELINK_HOSTNAME_NODE	string
SYSLOG_FACILITY	int
ZONE_LOCAL	string

optag Values for scha_control(1HA)
CHANGE_STATE_OFFLINE
CHANGE_STATE_ONLINE
CHECK_GIVEOVER
CHECK_RESTART
GIVEOVER
IGNORE_FAILED_START
RESOURCE_DISABLE
RESOURCE_IS_RESTARTED
RESOURCE_RESTART
RESTART

optag Values for scha_resource_get(1HA)	Result Type
AFFINITY_TIMEOUT	int
ALL_EXTENSIONS	string_array
BOOT_TIMEOUT	int
CHEAP_PROBE_INTERVAL	int

optag Values for scha_resource_get(1HA)	Result Type
CHEAP_PROBE_INTERVAL	int
EXTENSION	extension
EXTENSION_NODE	extension
FAILOVER_MODE	enum (NONE, HARD, SOFT, RESTART_ONLY, LOG_ONLY)
FINI_TIMEOUT	int
GROUP	string
INIT_TIMEOUT	int
LOAD_BALANCING_POLICY	string
LOAD_BALANCING_WEIGHTS	string_array
LOGICAL_HOSTNAMES_USED	string_array
MONITORED_SWITCH	enum (DISABLED, ENABLED)
MONITORED_SWITCH_NODE	enum (DISABLED, ENABLED)
MONITOR_CHECK_TIMEOUT	int
MONITOR_START_TIMEOUT	int
MONITOR_STOP_TIMEOUT	int
NETWORK_RESOURCES_USED	string_array
NUM_RESOURCE_RESTARTS	int
NUM_RESOURCE_RESTARTS_ZONE	int
NUM_RG_RESTARTS	int
NUM_RG_RESTARTS	int
NUM_RG_RESTARTS_ZONE	int
ON_OFF_SWITCH	enum (DISABLED, ENABLED)
ON_OFF_SWITCH_NODE	enum (DISABLED, ENABLED)
PORT_LIST	string_array
POSTNET_STOP_TIMEOUT	int
PRENET_START_TIMEOUT	int
R_DESCRIPTION	string
RESOURCE_DEPENDENCIES	string_array

optag Values for scha_resource_get(1HA)	Result Type
RESOURCE_DEPENDENCIES_OFFLINE_RESTART	string_array
RESOURCE_DEPENDENCIES_RESTART	string_array
RESOURCE_DEPENDENCIES_WEAK	string_array
RESOURCE_PROJECT_NAME	string
RESOURCE_STATE	enum (ONLINE, OFFLINE, START_FAILED, STOP_FAILED, MONITOR_FAILED, ONLINE_NOT_MONITORED, STARTING, STOPPING)
RESOURCE_STATE_NODE	enum (see RESOURCE_STATE for values)
RETRY_COUNT	int
RETRY_INTERVAL	int
SCALABLE	boolean
START_TIMEOUT	int
STATUS	status
STATUS_NODE	status
STOP_TIMEOUT	int
THOROUGH_PROBE_INTERVAL	int
TYPE	string
TYPE_VERSION	string
UDP_AFFINITY	boolean
UPDATE_TIMEOUT	int
VALIDATE_TIMEOUT	int
WEAK_AFFINITY	boolean

optag Values for scha_resource_get(1HA) and scha_resourcetype_get(1HA)	Result Type
API_VERSION	int
BOOT	string
FAILOVER	boolean
FINI	string
GLOBAL_ZONE	boolean

optag **Values for** `scha_resource_get`**(1HA) and** `scha_resourcetype_get`**(1HA)**	**Result Type**
`INIT`	`string`
`INIT_NODES`	`enum` (`RG_PRIMARIES`, `RT_INSTALLED_NODES`)
`INSTALLED_NODES`	`string_array`. An asterisk (*) is returned to indicate all nodes.
`IS_LOGICAL_HOSTNAME`	`boolean`
`IS_SHARED_ADDRESS`	`boolean`
`MONITOR_CHECK`	`string`
`MONITOR_START`	`string`
`MONITOR_STOP`	`string`
`PER_NODE`	`boolean`
`PKGLIST`	`string_array`
`POSTNET_STOP`	`string`
`PRENET_START`	`string`
`PROXY`	`boolean`
`RT_BASEDIR`	`string`
`RT_DESCRIPTION`	`string`
`RT_SYSTEM`	`boolean`
`RT_VERSION`	`string`
`SINGLE_INSTANCE`	`boolean`
`START`	`string`
`STOP`	`string`
`UPDATE`	`string`
`VALIDATE`	`string`

optag **Values for** `scha_resourcegroup_get`**(1HA)**	**Result Type**
`AUTO_START_ON_NEW_CLUSTER`	`boolean`
`DESIRED_PRIMARIES`	`int`
`FAILBACK`	`boolean`

optag **Values for** `scha_resourcegroup_get`**(1HA)**	**Result Type**
`GLOBAL_RESOURCES_USED`	`string_array` (an asterisk (*) is returned to indicate all resources)
`IMPLICIT_NETWORK_DEPENDENCIES`	`boolean`
`MAXIMUM_PRIMARIES`	`int`
`NODELIST`	`string_array`
`PATHPREFIX`	`string`
`PINGPONG_INTERVAL`	`int`
`RESOURCE_LIST`	`string_array`
`RG_AFFINITIES`	`string_array`
`RG_DEPENDENCIES`	`string_array`
`RG_DESCRIPTION`	`string`
`RG_IS_FROZEN`	`boolean`
`RG_MODE`	`enum` (`FAILOVER`, `SCALABLE`)
`RG_PROJECT_NAME`	`string`
`RG_SLM_CPU`	`decimal`
`RG_SLM_CPU_MIN`	`decimal`
`RG_SLM_PSET_TYPE`	`enum` (`DEFAULT`, `DEDICATED_STRONG`, `DEDICATED_WEAK`)
`RG_SLM_TYPE`	`enum` (`AUTOMATED`, `MANUAL`)
`RG_STATE`	`enum` (`UNMANAGED`, `ONLINE`, `OFFLINE`, `PENDING_ONLINE`, `PENDING_OFFLINE`, `ERROR_STOP_FAILED`, `ONLINE_FAULTED`, `PENDING_ONLINE_BLOCKED`)
`RG_STATE_NODE`	`enum` (see `RG_STATE` for values)
`RG_SYSTEM`	`boolean`
`SUSPEND_AUTOMATIC_RECOVERY`	`boolean`

Exit Status One set of exit status codes is used for all `scha` commands.

The exit status codes are the numeric values of the `scha_err_t` return codes of the corresponding C functions as described in scha_calls(3HA).

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Stable

See Also `awk`(1), rt_callbacks(1HA), scha_cluster_get(1HA), scha_control(1HA), scha_resource_get(1HA), scha_resourcegroup_get(1HA), scha_resourcetype_get(1HA), scha_resource_setstatus(1HA), scha_calls(3HA), `attributes`(5), `formats`(5), r_properties(5), rg_properties(5), rt_properties(5)

Name scha_control – request resource and resource group control

Synopsis

```
scha_control  -O optag -G group -R resource [-Z zonename]
```

Description The `scha_control` command requests the restart or relocation of a resource or resource group that is under the control of the Resource Group Manager (RGM). Use this command in shell script implementations of resource monitors. This command provides the same functionality as the scha_control(3HA) C function.

The exit code of this command indicates whether the requested action was rejected. If the request is accepted, this command does not return until the resource group or resource has completed going offline and has come back online. The fault monitor that called scha_control(1HA) might be stopped as a result of the resource or resource group's going offline. As a result, the fault monitor might never receive the return status of a successful request.

You need `solaris.cluster.resource.admin` role-based access control (RBAC) authorization to use this command. See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`(1), `pfcsh`(1), or `pfksh`(1) profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su`(1M) to assume a role. You can also use `pfexec`(1) to issue privileged Sun Cluster commands.

Options The following options are supported:

`-G` *group*
: The name of the resource group that is to be restarted or relocated or that contains the resource that is to be restarted or relocated. If the resource group is not online on the node or zone where the request is made, the request is rejected.

`-O` *optag*
: Requests *optag* options.

  Note – *optag* options, such as `CHECK_GIVEOVER` and `CHECK_RESTART`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify *optag* options.

  The following *optag* values are supported:

  `CHANGE_STATE_OFFLINE`
  : Requests that the proxy resource that is named by the `-R` option be brought offline on the local node or zone. A *proxy resource* is a Sun Cluster resource that imports the state of a resource from another cluster such as Oracle Cluster Ready Services (CRS), now known as Oracle clusterware CRS. This change in state reflects, in the context of the Sun Cluster software, the change in state of the external resource.

When you change the state of a proxy resource with this *optag* value, methods of the proxy resource are not executed.

If a fault occurs on a "depended-on" resource on a node or zone, and the resource cannot recover, the monitor brings that resource on that node or zone offline. The monitor brings the resource offline by calling the `scha_control` command with the `CHANGE_STATE_OFFLINE` *optag* value. The monitor also brings all of the depended-on resource's offline-restart dependents offline by triggering a restart on them. When the cluster administrator resolves the fault and reenables the depended-on resource, the monitor brings the depended-on resource's offline-restart dependents back online as well.

`CHANGE_STATE_ONLINE`
Requests that the proxy resource that is named by the `-R` option be brought online on the local node or zone. A *proxy resource* is a Sun Cluster resource that imports the state of a resource from another cluster such as Oracle Cluster Ready Services (CRS). This change in state reflects, in the context of the Sun Cluster software, the change in state of the external resource.

When you change the state of a proxy resource with this *optag* value, methods of the proxy resource are not executed.

`CHECK_GIVEOVER`
Performs all the same validity checks that would be done for a `GIVEOVER` of the resource group that is named by the `-G` option, but does not actually relocate the resource group.

`CHECK_RESTART`
Performs all the same validity checks that would be done for a `RESTART` of the resource group that is named by the `-G` option, but does not actually restart the resource group.

`GIVEOVER`
Requests that the resource group that is named by the `-G` option be brought offline on the local node or zone, and online again on a different node or zone of the RGM's choosing. Note that if the resource group is currently online on two or more nodes or zones and there are no additional available nodes or zones on which to bring the resource group online, it can be taken offline on the local node or zone without being brought online elsewhere. The request might be rejected depending on the result of various checks. For example, a node or zone might be rejected as a host because the group was brought offline due to a `GIVEOVER` request on that node or zone within the interval specified by the `PINGPONG_INTERVAL` property.

If the cluster administrator configures the `RG_Affinities` properties of one or more resource groups, and you issue a **`scha_control GIVEOVER`** request on one resource group, more than one resource group might be relocated as a result. The `RG_Affinities` property is described in rg_properties(5).

The MONITOR_CHECK method is called before the resource group that contains the resource is relocated to a new node or zone as the result of a scha_control command or scha_control() function call from a fault monitor.

You can call the MONITOR_CHECK method on any node or zone that is a potential new master for the resource group. The MONITOR_CHECK method is intended to assess whether a node or zone is healthy enough to run a resource. The MONITOR_CHECK method must be implemented in such a way that it does not conflict with the running of another method concurrently.

MONITOR_CHECK failure vetoes the relocation of the resource group to the node or zone where the callback was invoked.

IGNORE_FAILED_START
Requests that if the currently executing Prenet_start or Start method fails, the resource group is not to fail over, regardless of the setting of the Failover_mode property.

In other words, this *optag* value overrides the recovery action that is normally taken for a resource for which the Failover_Mode property is set to SOFT or HARD when that resource fails to start. Normally, the resource group fails over to a different node or zone. Instead, the resource behaves as if Failover_Mode is set to NONE. The resource enters the START_FAILED state, and the resource group ends up in the ONLINE_FAULTED state, if no other errors occur.

This optag value is meaningful only when it is called from a Start or Prenet_start method that subsequently exits with a nonzero status or times out. This optag value is valid only for the current invocation of the Start or Prenet_start method. The scha_control command should be called with this optag value in a situation in which the Start method has determined that the resource cannot start successfully on another node or zone. If this optag value is called by any other method, the error SCHA_ERR_INVAL is returned. This optag value prevents the "ping pong" failover of the resource group that would otherwise occur.

RESOURCE_DISABLE
Disables the resource that is named by the -R option on the node or zone on which the scha_control command is called.

If a fault occurs on a "depended-on" resource on a node or zone, and the resource cannot recover, the monitor brings that resource on that node or zone offline. The monitor brings the resource offline by calling the scha_control command with the RESOURCE_DISABLE *optag* value. The monitor also brings all of the depended-on resource's offline-restart dependents offline by triggering a restart on them. When the cluster administrator resolves the fault and reenables the depended-on resource, the monitor brings the depended-on resource's offline-restart dependents back online as well.

RESOURCE_IS_RESTARTED
Requests that the resource restart counter for the resource that is named by the -R option be incremented on the local node or zone, without actually restarting the resource.

A resource monitor that restarts a resource directly without calling the RESOURCE_RESTART option of scha_control (for example, using pmfadm(1M)) can use this option to notify the RGM that the resource has been restarted. This incrementing is reflected in subsequent NUM_RESOURCE_RESTARTS queries of scha_resource_get(1HA).

If the resource's type fails to declare the RETRY_INTERVAL standard property, the RESOURCE_IS_RESTARTED option of the scha_control command is not permitted. Consequently, the scha_control command fails and generates exit status code 13 (SCHA_ERR_RT).

RESOURCE_RESTART
Requests that the resource that is named by the -R option be brought offline and online again on the local node or zone without stopping any other resources in the resource group. The resource is stopped and started by applying the following sequence of methods to it on the local node or zone:

```
MONITOR_STOP
STOP
START
MONITOR_START
```

If the resource's type does not declare a MONITOR_STOP and MONITOR_START method, only the STOP and START methods are invoked to perform the restart. If the resource's type does not declare both a START and STOP method, the scha_control command fails and generates exit status code 13 (SCHA_ERR_RT).

If a method invocation fails while restarting the resource, the RGM might set an error state, relocate the resource group, or reboot the node or zone, depending on the setting of the FAILOVER_MODE property of the resource. For additional information, see the FAILOVER_MODE property in r_properties(5).

A resource monitor using this option to restart a resource can use the NUM_RESOURCE_RESTARTS query of scha_resource_get(1HA) to keep count of recent restart attempts.

The RESOURCE_RESTART function should be used with care by resource types that have PRENET_START, POSTNET_STOP, or both methods. Only the MONITOR_STOP, STOP, START, and MONITOR_START methods are applied to the resource. Network address resources on which this resource depends are not restarted and remain online.

If a fault occurs on a “depended-on” resource on a node or zone, and the resource cannot recover, the monitor brings that resource on that node or zone offline. The monitor brings the resource offline by calling the scha_control command with the

RESOURCE_RESTART *optag* value. The monitor also brings all of the depended-on resource's offline-restart dependents offline by triggering a restart on them. When the cluster administrator resolves the fault and reenables the depended-on resource, the monitor brings the depended-on resource's offline-restart dependents back online as well.

RESTART
Requests that the resource group that is named by the -G option be brought offline, then online again, without forcing relocation to a different node or zone. The request might ultimately result in relocating the resource group if a resource in the group fails to restart. A resource monitor using this option to restart a resource group can use the NUM_RG_RESTARTS query of scha_resource_get(1HA) to keep count of recent restart attempts.

The CHECK_GIVEOVER and CHECK_RESTART *optag* values are intended to be used by resource monitors that take direct action upon resources (for example, killing and restarting processes, or rebooting nodes or zones) rather than invoking the scha_control command to perform a giveover or restart. If the check fails, the monitor should sleep for awhile and restart its probes rather than invoke its restart or failover actions. For more information, see scha_control(3HA).

-R *resource*
The name of a resource in the resource group, presumably the resource whose monitor is making the scha_control(1HA) request. If the named resource is not in the resource group, the request is rejected.

The setting of the Failover_mode property of the indicated resource might suppress the requested scha_control action. If Failover_mode is RESTART_ONLY, all requests except scha_control GIVEOVER and scha_control CHECK_GIVEOVER are permitted. The GIVEOVER and CHECK_GIVEOVER requests return the SCHA_ERR_CHECKS exit code and the requested giveover action is not executed, producing only a syslog message.

If the Retry_count and Retry_interval properties are set on the resource, the number of resource restarts is limited to Retry_count attempts within the Retry_interval. If Failover_mode is LOG_ONLY, any scha_control giveover, restart, or disable request returns the SCHA_ERR_CHECKS exit code and the requested giveover or restart action is not executed, producing only a syslog message.

-Z *zonename*
The name of the zone in which a resource group is configured to run.

If the Global_zone property is set to TRUE, methods execute in the global zone even if the resource group that contains the resource runs in a non-global zone. This option provides the name of the non-global zone in which the resource group is configured to run.

Use the -Z option only for resource types whose Global_zone property is set to TRUE. This option is not needed if the Global_zone property is set to FALSE. For more information about the Global_zone property, see the rt_properties(5) man page.

Exit Status The following exit status codes are returned:

`0` The command completed successfully.

nonzero An error occurred.

Failure error codes are described in scha_calls(3HA).

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Stable

See Also pmfadm(1M), rt_callbacks(1HA), scha_cmds(1HA), scha_resource_get(1HA), scha_calls(3HA), scha_control(3HA), scha_control_zone(3HA), `attributes`(5), r_properties(5), `rbac`(5), rg_properties(5), rt_properties(5)

Name scha_resource_get – access resource information

Synopsis
```
scha_resource_get [-Q] -O optag -R resource [-G group]
 [args]
```

Description The `scha_resource_get` command accesses information about a resource that is under the control of the Resource Group Manager (RGM). You can use this command to query the properties of the resource's type, as described in rt_properties(5), as well as the properties of the resource, as described in r_properties(5).

Use the `scha_resource_get` command in shell script implementations of the callback methods for resource types that represent services that are controlled by the cluster's RGM. This command provides the same information as the scha_resource_get(3HA) C function.

Information is generated by the command to `stdout` in formatted strings on separate lines, as described in scha_cmds(1HA). The output can be stored in shell variables and parsed by using shell facilities or `awk`(1) for further use by the script.

You need `solaris.cluster.resource.read` role-based access control (RBAC) authorization to use this command. See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`(1), `pfcsh`(1), or `pfksh`(1) profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su`(1M) to assume a role. You can also use `pfexec`(1) to issue privileged Sun Cluster commands.

Options The following options are supported:

`-G` *group*
The name of the resource group in which the resource has been configured. Although this argument is optional, the command will run more efficiently if you include it.

`-O` *optag*
Indicates the information to be accessed. Depending on the *optag* value that you specify, you might need to include an additional value to indicate the cluster node or zone for which information is to be retrieved.

Note – *optag* values, such as `AFFINITY_TIMEOUT` and `BOOT_TIMEOUT`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify *optag* values.

The following *optag* values retrieve the corresponding resource properties. The value of the named property of the resource is generated. The `NUM_RG_RESTARTS`, `NUM_RESOURCE_RESTARTS`, `MONITORED_SWITCH`, `ON_OFF_SWITCH`, `RESOURCE_STATE`, and `STATUS` properties refer to the value on the node or zone where the command is executed.

See the r_properties(5) man page for descriptions of the resource properties that correspond to the following *optag* values. Note that some *optag* values in the following list are described after the list rather than in the r_properties(5) man page.

The `EXTENSION` property also refers to the value on the node or zone where the command is executed, provided that the `PER_NODE` property attribute is set. See the property_attributes(5) man page.

```
AFFINITY_TIMEOUT
ALL_EXTENSIONS
BOOT_TIMEOUT
CHEAP_PROBE_INTERVAL
EXTENSION
EXTENSION_NODE
FAILOVER_MODE
FINI_TIMEOUT
GROUP
INIT_TIMEOUT
LOAD_BALANCING_POLICY
LOAD_BALANCING_WEIGHTS
LOGICAL_HOSTNAMES_USED
MONITORED_SWITCH
MONITORED_SWITCH_NODE
MONITOR_CHECK_TIMEOUT
MONITOR_START_TIMEOUT
MONITOR_STOP_TIMEOUT
NETWORK_RESOURCES_USED
NUM_RESOURCE_RESTARTS
NUM_RG_RESTARTS
ON_OFF_SWITCH
ON_OFF_SWITCH_NODE
PORT_LIST
POSTNET_STOP_TIMEOUT
PRENET_START_TIMEOUT
RESOURCE_DEPENDENCIES
RESOURCE_DEPENDENCIES_OFFLINE_RESTART
RESOURCE_DEPENDENCIES_RESTART
RESOURCE_DEPENDENCIES_WEAK
RESOURCE_PROJECT_NAME
RESOURCE_STATE
RESOURCE_STATE_NODE
RETRY_COUNT
RETRY_INTERVAL
R_DESCRIPTION
SCALABLE
START_TIMEOUT
STATUS
STATUS_NODE
```

```
STOP_TIMEOUT
THOROUGH_PROBE_INTERVAL
TYPE
TYPE_VERSION
UDP_AFFINITY
UPDATE_TIMEOUT
VALIDATE_TIMEOUT
WEAK_AFFINITY
```

The following *optag* values are not described in the r_properties(5) man page.

`ALL_EXTENSIONS`
: Generates, on successive lines, the names of all extension properties of the resource.

`EXTENSION`
: Generates the type of property followed by its value, on successive lines. If the property is a per-node extension property, the value that is returned is the value of the property on the node on which `scha_resource_get` is executed. Requires an unflagged argument that names an extension property of the resource. Shell scripts might need to discard the type to obtain the value, as shown in EXAMPLES.

  When a user requests the value of this property on a node or zone for which an explicit value has not been assigned, the default value that is declared in the Resource Type Registration (RTR) file is returned. See the rt_reg(4) man page.

`EXTENSION_NODE`
: Generates the type of property followed by its value, on successive lines, for the named node or zone. This value requires two unflagged arguments, in the following order, that name an extension of the resource on a particular node or zone:

  - Extension property name
  - Node or zone name

  Shell scripts might need to discard the type to obtain the value.

  When a user requests the value of this property on a node or zone for which an explicit value has not been assigned, the default value that is declared in the RTR file is returned. See the rt_reg(4) man page.

`GROUP`
: Generates the name of the resource group in which the resource is configured.

`RESOURCE_STATE_NODE`
: Generates the value of the resource's `RESOURCE_STATE` property for the named node or zone. Requires an unflagged argument that names a node or zone.

`STATUS_NODE`
: Generates the value of the resource's `STATUS` property for the named node or zone. Requires an unflagged argument that names a node or zone.

The following *optag* values retrieve the corresponding resource type properties. The value of the named property of the resource's type is generated.

Note – *optag* values, such as `API_VERSION` and `BOOT`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify *optag* values.

For descriptions of resource type properties, see rt_properties(5).

```
API_VERSION
BOOT
FAILOVER
FINI
INIT
INIT_NODES
INSTALLED_NODES
IS_LOGICAL_HOSTNAME
IS_SHARED_ADDRESS
MONITOR_CHECK
MONITOR_START
MONITOR_STOP
PKGLIST
POSTNET_STOP
PRENET_START
RT_BASEDIR
RT_DESCRIPTION
RT_SYSTEM
RT_VERSION
SINGLE_INSTANCE
START
STOP
UPDATE
VALIDATE
```

`-Q`
Include any specified qualifiers in resource dependency lists. The `{LOCAL_NODE}`, `{ANY_NODE}`, and `{FROM_RG_AFFINITIES}` qualifiers are described in the r_properties(5) man page. If you omit the `-Q` option, the returned value of resource dependency lists contain only the resource names, without any declared qualifiers.

`-R` *resource*
The name of a resource that is being managed by the RGM cluster facility.

Examples

EXAMPLE 1 Sample Script That Uses the `scha_resource_get` Command

The following script is passed `-R` and `-G` arguments, which provide the required resource name and resource group name. Next, the `scha_resource_get` command accesses the `Retry_count` property of the resource and the `enum`-type `LogLevel` extension property of the resource.

EXAMPLE 1 Sample Script That Uses the `scha_resource_get` Command *(Continued)*

```
#!/bin/sh

while getopts R:G: opt
do
     case $opt in
           R)      resource="$OPTARG";;
           G)      group="$OPTARG";;
     esac
done

retry_count=‘scha_resource_get -O Retry_count -R $resource \\
-G $group‘
printf "retry count for resource %s is %d\n" $resource \\
$retry_count

LogLevel_info=‘scha_resource_get -O Extension -R $resource \\
-G $group LogLevel‘

# Get the enum value that follows the type information
# of the extension property.  Note that the preceding
# assignment has already changed the newlines separating
# the type and the value to spaces for parsing by awk.

loglevel=‘echo $LogLevel_info | awk ’{print $2}’‘
```

EXAMPLE 2 Using the `scha_resource_get` Command With and Without the `-Q` Option to Query Resource Dependencies

This example shows how to use the `clresource` command to create a resource named `myres`, with several resource dependencies that have a `{LOCAL_NODE}` scope qualifier, a `{ANY_NODE}` scope qualifier, or no scope qualifier. This example then shows how to use the `scha_resource_get` command to query the `Resource_dependencies` property. Without the `-Q` option, only resource names are returned. With the `-Q` option, the declared scope qualifiers are returned as well.

```
# clresource create -g mygrp -t myrestype \
-p Resource_dependencies=myres2{LOCAL_NODE},myres3{ANY_NODE},myres4 \
myres
# scha_resource_get -O Resource_dependencies -R myres -G mygrp
myres2
myres3
myres4
# scha_resource_get -Q -O Resource_dependencies -R myres -G mygrp
myres2{LOCAL_NODE}
myres3{ANY_NODE}
myres4
```

EXAMPLE 2 Using the `scha_resource_get` Command With and Without the `-Q` Option to Query Resource Dependencies *(Continued)*

```
#
```

Exit Status

The following exit status codes are returned:

`0` The command completed successfully.

nonzero An error occurred.

Failure error codes are described in scha_calls(3HA).

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also

`awk`(1), scha_cmds(1HA), scha_calls(3HA), scha_resource_get(3HA), rt_reg(4), `attributes`(5), property_attributes(5), r_properties(5), rt_properties(5)

Name scha_resourcegroup_get – access resource group information

Synopsis

```
scha_resourcegroup_get  -O optag -G group [args]
```

Description The `scha_resourcegroup_get` command accesses information about a resource group that is under the control of the Resource Group Manager (RGM) cluster facility.

This command is intended to be used in shell script implementations of the callback methods for resource types. These resource types represent services that are controlled by the cluster's RGM. This command provides the same information as the scha_resourcegroup_get(3HA) C function.

Information is generated by the command to standard output (`stdout`) in formatted strings as described in scha_cmds(1HA). The output is a string or several strings on separate lines. The output can be stored in shell variables and parsed using shell facilities or `awk`(1) for further use by the script.

You need `solaris.cluster.resource.read` role-based access control (RBAC) authorization to use this command. See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`(1), `pfcsh`(1), or `pfksh`(1) profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su`(1M) to assume a role. You can also use `pfexec`(1) to issue privileged Sun Cluster commands.

Options The following options are supported:

`-G` *group* Name of the resource group.

`-O` *optag* Specifies the information that is to be accessed. Depending on the *optag* that you specify, you might need to include an additional operand to indicate the node or zone for which information is to be retrieved.

Note – *optag* values, such as `DESIRED_PRIMARIES` and `FAILBACK`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify *optag* options.

The following *optag* values retrieve the corresponding resource group properties. The value of the named property of the resource group is generated. The `RG_STATE` property refers to the value on the particular node or zone where the command is executed.

```
AUTO_START_ON_NEW_CLUSTER
DESIRED_PRIMARIES
FAILBACK
GLOBAL_RESOURCES_USED
```

```
IMPLICIT_NETWORK_DEPENDENCIES
MAXIMUM_PRIMARIES
NODELIST
PATHPREFIX
PINGPONG_INTERVAL
RESOURCE_LIST
RG_AFFINITIES
RG_DEPENDENCIES
RG_DESCRIPTION
RG_IS_FROZEN
RG_MODE
RG_PROJECT_NAME
RG_SLM_TYPE
RG_SLM_PSET_TYPE
RG_SLM_CPU
RG_SLM_CPU_MIN
RG_STATE
RG_STATE_NODE
RG_SYSTEM
SUSPEND_AUTOMATIC_RECOVERY
```

Note – `RG_STATE_NODE` requires an unflagged argument that specifies a node or zone. This *optag* value generates the value of the resource group's `RG_STATE` property for the specified node or zone. If the unflagged argument specifies a non-global zone, the format is *nodename*:*zonename*.

Examples

EXAMPLE 1 A Sample Script Using `scha_resourcegroup_get`

The following script is passed a `-G` argument, which provides the required resource group name. Next, the `scha_resourcegroup_get` command is used to get the list of resources in the resource group.

```
#!/bin/sh

while getopts G: opt
do
     case $opt in
          G)      group="$OPTARG";;
     esac
done

resource_list=‘scha_resourcegroup_get -O Resource_list -G $group‘

for resource in $resource_list
do
     printf "Group: %s contains resource: %s\n" "$group" "$resource"
done
```

Exit Status The following exit status codes are returned:

`0` The command completed successfully.

nonzero An error occurred.

Failure error codes are described scha_calls(3HA).

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Stable

See Also `awk`(1), scha_cmds(1HA), scha_calls(3HA), scha_resourcegroup_get(3HA), `attributes`(5), rg_properties(5), `rbac`(5)

Name scha_resource_setstatus – set resource status

Synopsis
```
scha_resource_setstatus  -R resource -G group -s status
 [-m msg] [-Z zonename]
```

Description The `scha_resource_setstatus` command sets the `Status` and `Status_msg` properties of a resource that is managed by the Resource Group Manager (RGM). This command is intended to be used by the resource's monitor to indicate the resource's state as perceived by the monitor. It provides the same functionality as the scha_resource_setstatus(3HA) C function.

When you execute the scha_resource_setstatus(1HA) command, the `Status` and `Status_msg` properties of the resource are updated with the values that you specify. Sun Cluster logs the change to the resource status in the cluster system log, which you can view with cluster administration tools.

You need `solaris.cluster.resource.admin` RBAC authorization to use this command. See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`(1), `pfcsh`(1), or `pfksh`(1) profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su`(1M) to assume a role. You can also use `pfexec`(1) to issue privileged Sun Cluster commands.

Options The following options are supported:

`-G` *group*
Specifies the resource group that contains the resource.

`-m` *msg*
Specifies the text string that you want to assign to the `Status_msg` property of the resource. If you do not specify this option, the value of the resource's `Status_msg` is set to `NULL`.

`-R` *resource*
Specifies the resource whose status is to be set.

`-s` *status*
Specifies the value of *status*: `OK`, `DEGRADED`, `FAULTED`, `UNKNOWN`, or `OFFLINE`.

`-Z` *zonename*
Specifies the name of the non-global zone in which a resource group is configured to run.

If the `Global_zone` property is set to `TRUE`, methods execute in the global zone even if the resource group that contains the resource runs in a non-global zone. The `-Z` option sets the status for the non-global zone where the resource group runs rather than for the global zone where the method runs.

Use the -Z option only for resource types whose Global_zone property is set to TRUE. This option is not needed if the Global_zone property is set to FALSE. For more information about the Global_zone property, see the rt_properties(5) man page.

Examples

EXAMPLE 1 Setting the Status of Resource R1 With a Status_msg

The following command sets the status of resource R1 in resource group RG2 to OK and sets the Status_msg to Resource R1 is OK:

```
scha_resource_setstatus -R R1 -G RG2 -s OK -m "Resource R1 is OK"
```

EXAMPLE 2 Setting the Status of Resource R1 Without a Status_msg

The following command sets the status of R1 in resource group RG2 to DEGRADED and sets the Status_msg to NULL:

```
scha_resource_setstatus -R R1 -G RG2 -s DEGRADED
```

EXAMPLE 3 Setting the Status of Resource R1 in Zone Zone1 With a Status_msg

The following example shows a resource type method or monitor that is implemented as a shell script. This shell script shows how to set the status of resource $resource in resource group $rg in zone $localzone to OK. This shell script also sets the Status_msg to "Resource R1 is OK". In this case, the -Z option must be specified because the resource type property Global_zone is assumed to be set to TRUE.

```
resource=
rg=""
localzone=""
zflag=""
while getopts R:G:Z:
do
      case $c in
      R)   resource=$OPTARG;;
      G)   rg=$OPTARG;;
      Z)   zflag="-Z"
           localzone=$OPTARG;;
      esac
done
...
scha_resource_setstatus -R $resource -G $rg $zflag $localzone -s OK -m "Resource R1 is OK"
```

Exit Status

The following exit status codes are returned:

0	The command completed successfully.
nonzero	An error occurred.
	Failure error codes are described in scha_calls(3HA).

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Stable

See Also scha_cmds(1HA), scha_calls(3HA), scha_resource_setstatus(3HA), `attributes`(5), `rbac`(5), rt_properties(5)

Name scha_resourcetype_get – access resource type information

Synopsis

```
scha_resourcetype_get  -O optag -T type
```

Description The `scha_resourcetype_get` command accesses information about a resource type that is registered with the Resource Group Manager (RGM).

Use this command in shell script implementations of the callback methods for resource types that represent services that are controlled by the RGM. This command provides the same information as the scha_resourcetype_get(3HA) C function.

Information is output by this command to the standard output (`stdout`) in formatted strings, as described in the scha_cmds(1HA) man page. Output is a string or several strings that are output on separate lines. You can store the output in shell variables. You can also parse the output by using the `awk`(1) command or other shell commands for further use by the script.

You need `solaris.cluster.resource.read` RBAC authorization to use this command. See the `rbac`(5) man page.

Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`(1), `pfcsh`(1), or `pfksh`(1) profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su`(1M) to assume a role. You can also use `pfexec`(1) to issue privileged Sun Cluster commands.

Options The following options are supported:

`-O` *optag* Indicates the information to be accessed.

Note – *optag* options, such as `API_VERSION` and `BOOT`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify *optag* options.

The following *optag* values retrieve the corresponding resource type properties. The value of the named property of the resource's type is output.

```
API_VERSION
BOOT
FAILOVER
FINI
INIT
INIT_NODES
INSTALLED_NODES
IS_LOGICAL_HOSTNAME
IS_SHARED_ADDRESS
MONITOR_CHECK
MONITOR_START
MONITOR_STOP
PKGLIST
```

```
POSTNET_STOP
PRENET_START
RESOURCE_LIST
RT_BASEDIR
RT_DESCRIPTION
RT_SYSTEM
RT_VERSION
SINGLE_INSTANCE
START
STOP
UPDATE
VALIDATE
```

`-T` *type* Is the name of a resource type that is registered for use by the RGM cluster facility.

Exit Status The following exit values are returned:

`0` The command completed successfully.

nonzero An error occurred.

Failure error codes are described scha_calls(3HA).

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Stable

See Also `awk`(1), scha_cmds(1HA), scha_calls(3HA), scha_resourcetype_get(3HA), `attributes`(5), rt_properties(5)

REFERENCE

SC32 1m

Name cconsole , crlogin, cssh, ctelnet – multi window, multi machine, remote console, login, ssh, and telnet commands

Synopsis

```
$CLUSTER_HOME/bin/cconsole [-s] [ -l  username] [clustername... | nodename...]

$CLUSTER_HOME/bin/crlogin [-l username] [clustername...
 | nodename...]

$CLUSTER_HOME/bin/cssh [-l username] [-p ssh-port] [clustername...
 | nodename...]

$CLUSTER_HOME/bin/ctelnet [clustername... | nodename...]
```

Description These utilities initiate a multiple window connection to a set of specified hosts. There are three variations: one that is specifically intended for remote console access while the others provide remote logins using `rlogin`, `ssh`, or `telnet`.

Each utility starts a host window for each of the specified hosts, as well as a common window. Input directed into the common window is sent to each of these host windows.

This tool is useful for system administration tasks that require similar things to be done on each of several hosts. For tasks that are identical on all hosts, typing in the common window sends the characters to all of the hosts. However, the host windows are normal terminal windows so they can also be used one at a time (by moving the mouse into one of them and typing directly into it) to perform host specific tasks.

The common window also allows the user to select which hosts receive the characters typed in the common window, so only the specified hosts will receive input.

These utilities use entries in two different databases, `/etc/clusters` and `/etc/serialports`.

cconsole The `cconsole` command provides remote console access through the `telnet` command. All normal `telnet` escape characters are available to the user. See the `telnet`(1) man page for a complete listing of `telnet` escape characters. Because there are a few `telnet` escapes that are commonly used, they are provided here as well. The escape character is Control-], specified below as ^].

^] quit	Quit the session. Analogous to `~.` in `tip` and `rlogin`.
^] send brk	Send a break signal to the remote system. This is what is needed to halt the Sun CPU. The normal key board sequence is “L1-A.”

crlogin The `crlogin` utility provides remote login through the `rlogin` command to multiple cluster nodes.

cssh The `cssh` utility establishes Secure Shell connections directly to the cluster nodes through the `ssh` command.

ctelnet The `ctelnet` utility is similar to `cconsole` except that the connection is directly over the Internet.

Options The following options are supported:

`-l` *username* Specifies the user name for the `ssh` connection. This option is passed to the `cconsole`, `crlogin`, or `cssh` utility when the utility is launched from the CCP. The `ctelnet` utility ignores this option.

If the `-l` option is not specified, the user name that launched the CCP is effective.

`-p` *ssh-port* Specifies the Secure Shell port number to use. This option is passed to the `cssh` utility when the utility is launched from the CCP. The `cconsole`, `crlogin`, and `ctelnet` utilities ignore this option,

If the `-p` option is not specified, the default port number 22 is used for secure connections.

`-s` Specifies using Secure Shell connections to node consoles instead of `telnet` connections. This option is passed to the `cconsole` utility when the utility is launched from the CCP. The `crlogin`, `cssh`, and `ctelnet` utilities ignore this option.

If the `-s` option is not specified, the cconsole utility uses `telnet` connections to the consoles.

To override the `-s` option, deselect the Use SSH checkbox in the Options menu of the `cconsole` graphical user interface (GUI).

Environment Variables The following environment variables affect the execution of these utilities:

`CLUSTER_HOME` Location of Sun Cluster System tools. Defaults to `/opt/SUNWcluster`.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWccon
Interface Stability	Evolving

See Also `rlogin`(1), `ssh`(1), `telnet`(1), `tip`(1), ccp(1M), chosts(1M), cports(1M), clusters(4), serialports(4), `attributes`(5)

Notes The standard set of X Window System command line arguments are accepted.

Name ccp – the Sun Cluster System Cluster Control Panel GUI

Synopsis `$CLUSTER_HOME/bin/ccp [-s] [ -l` *username*`] [-p` *ssh-port*`][`*clustername* | *nodename*`]`

Description The `ccp` utility is a launch pad for the `cconsole`, `crlogin`, `cssh`, and `ctelnet` cluster utilities.

The `ccp` utility also accepts the standard set of X Window System command line arguments.

Options The following options are supported:

`-l` *username* — Specifies the user name for the `ssh` connection. This option is passed to the `cconsole`, `crlogin`, or `cssh` utility when the utility is launched from the CCP. The `ctelnet` utility ignores this option.

If the `-l` option is not specified, the user name that launched the CCP is effective.

`-p` *ssh-port* — Specifies the Secure Shell port number to use. This option is passed to the `cssh` utility when the utility is launched from the CCP. The `cconsole`, `crlogin`, and `ctelnet` utilities ignore this option,

If the `-p` option is not specified, the default port number 22 is used for secure connections.

`-s` — Specifies using Secure Shell connections to node consoles instead of `telnet` connections. This option is passed to the `cconsole` utility when the utility is launched from the CCP. The `crlogin`, `cssh`, and `ctelnet` utilities ignore this option.

If the `-s` option is not specified, the cconsole utility uses `telnet` connections to the consoles.

To override the `-s` option, deselect the Use SSH checkbox in the Options menu of the `cconsole` graphical user interface (GUI).

Operands The following operands are supported:

clustername — If provided, this option can be passed on as an argument to a tool in CCP's set of tools. The *clustername* argument can be specified by adding `$CLUSTER` in a tool's command line property.

Environment Variables The following environment variables affect the execution of the `ccp` utility:

`CLUSTER_HOME` — Location of cluster tools. Defaults to `/opt/SUNWcluster`.

`CCP_CONFIG_DIR` — Location of the tools' configuration files containing tool properties. Defaults to `/opt/SUNWcluster/etc/ccp`.

Files `$CLUSTER_HOME/etc/ccp/*`

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWccon
Interface Stability	Evolving

See Also `ssh`(1), cconsole(1M), chosts(1M), cports(1M), crlogin(1M), cssh(1M), ctelnet(1M), serialports(4), `attributes`(5)

Name chosts – expand cluster names into host names

Synopsis

```
$CLUSTER_HOME/bin/chosts name [name...]
```

Description The `chosts` utility expands the arguments into a list of host names.

Operands The following operands are supported:

name The parameter *name* can be a hostname or a cluster name. If *name* is a hostname, it is expanded to be a hostname. If *name* is a cluster name, that is, an entry exists in the `/etc/clusters` database (or a NIS or NIS+ map), it is expanded into the list of hosts that make up that cluster, as specified in the database. The list is typically used by programs that wish to operate on a list of hosts.

If an entry for `clusters` has been made in the `/etc/nisswitch.conf` file, then the order of lookups is controlled by that entry. If there is no such file or no such entry, then the nameservice look up order is implicitly `nis files`.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWccon
Interface Stability	Evolving

See Also ccp(1M), cconsole(1M), cports(1M), crlogin(1M), cssh(1M), ctelnet(1M), clusters(4), `attributes`(5)

Name cl_eventd – Cluster event daemon

Synopsis

```
/usr/cluster/lib/sc/cl_eventd [-v]
```

Description The `cl_eventd` daemon is started at boot time to monitor system events that are generated by other cluster components. This daemon also forwards these events to other cluster nodes. Only the events of class `EC_Cluster` are forwarded to other cluster nodes.

Options The following option is supported:

`-v` Send additional troubleshooting and debugging information to `syslogd`(1M).

Files `/usr/cluster/lib/sc/cl_eventd` Cluster event daemon

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu

See Also `syseventd`(1M), `syslog`(3C)

Notes The `cl_eventd` daemon does not provide a publicly accessible interface.

Name cports – expand host names into <host, server, port> triples

Synopsis

```
$CLUSTER_HOME/bin/cports hostname [hostname...]
```

Description The `cports` utility expands the *hostname* arguments into a list of `<host, server, port>` triples. The returned information is used to access the serial port consoles of the named hosts by way of the terminal server returned in the triples.

If an entry for `serialports` has been made in the `/etc/nisswitch.conf` file, then the order of lookups is controlled by that entry. If there is no such file or no such entry, then the nameservice look up order is implicitly `nis files`.

Examples **EXAMPLE 1** Using the `cports` Command

If the `/etc/serialports` file contains the entry:

```
pepsi    soda-tc    5002
```

the following command:

```
% cports pepsi
```

prints the following string:

```
pepsi soda-tc 5002
```

This information can be used by the `telnet`(1) command to remotely access `pepsi`'s console:

```
% telnet soda-tc 5002
```

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWccon
Interface Stability	Evolving

See Also `telnet`(1), cconsole(1M), ccp(1M), chosts(1M), crlogin(1M), cssh(1M), ctelnet(1M), serialports(4), `attributes`(5)

Name halockrun – run a child program while holding a file lock

Synopsis

```
/usr/cluster/bin/halockrun [-nsv] [-e exitcode] lockfilename
 prog [args]
```

Description The `halockrun` utility provides a convenient means to claim a file lock on a file and run a program while holding that lock. As this utility supports script locking, this utility is useful when programming in scripting languages such as the Bourne shell. See `sh`(1).

`halockrun` opens the file *lockfilename* and claims an exclusive mode file lock on the entire file. See `fcntl`(2) `fcntl`(2)). Then it runs the program *prog* with arguments *args* as a child process and waits for the child process to exit. When the child exits, `halockrun` releases the lock, and exits with the same exit code with which the child exited.

The overall effect is that the child *prog* is run as a critical section, and that this critical section is well-formed, in that no matter how the child terminates, the lock is released.

If the file *lockfilename* cannot be opened or created, then `halockrun` prints an error message on stderr and exits with exit code `99`.

You can run this command in the global zone or in a non-global zone. The command affects only the global or non-global zone in which you issue the command.

Options The following options are supported:

`-e` *exitcode*
: Normally, errors detected by `halockrun` exit with exit code `99`. The `-e` option provides a means to change this special exit code to a different value.

`-n`
: The lock should be requested in non-blocking mode: if the lock cannot be granted immediately, `halockrun` exits immediately, with exit code `1`, without running *prog*. This behavior is not affected by the `-e` option.

  Without the `-n` option, the lock is requested in blocking mode, thus, the `halockrun` utility blocks waiting for the lock to become available.

`-s`
: Claim the file lock in shared mode, rather than in exclusive mode.

`-v`
: Verbose output, on `stderr`.

Exit Status Errors detected by `halockrun` itself, such that the child process was never started, cause `halockrun` to exit with exit code `99`. (This exit code value can be changed to a different value using the `-e` option. See `OPTIONS`.

Otherwise, `halockrun` exits with the same exit code with which the child exited.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also `fcntl`(2), `attributes`(5)

Name hatimerun – run child program under a timeout

Synopsis

```
/usr/cluster/bin/hatimerun -t timeOutSecs [-av] [-e exitcode]
 prog args

/usr/cluster/bin/hatimerun -t timeOutSecs [-v] [-e exitcode]
 [-k signalname] prog args
```

Description The `hatimerun` utility provides a convenient facility for timing out the execution of another child, program. It is useful when programming in scripting languages, such as the Bourne shell. See `sh`(1).

The `hatimerun` utility runs the program *prog* with arguments *args* as a child subprocess under a timeout, and as its own process group. The timeout is specified in seconds, by the `-t` *timeOutSecs* option. If the timeout expires, then `hatimerun` kills the child subprocess's process group with a SIGKILL signal, and then exits with exit code 99.

You can run this command in the global zone or in a non-global zone. The command affects only the global or non-global zone in which you issue the command.

Options The following options are supported:

`-a` Changes the meaning of `hatimerun` radically: instead of killing the child when the timeout expires, the `hatimerun` utility simply exits, with exit code 99, leaving the child to run asynchronously.

It is illegal to supply both the `-a` option and the `-k` option.

`-e` Changes the exit code for the timeout case to some other value than 99.

`-k` Specifies what signal is used to kill the child process group. The possible signal names are the same as those recognized by the `kill`(1) command. In particular, the signal name should be one of the symbolic names defined in the `<signal.h>` description. The signal name is recognized in a case-independent fashion, without the `SIG` prefix. It is also legal to supply a numeric argument to the `-k` option, in which case that signal number is used.

It is illegal to supply both the `-a` option and the `-k` option.

`-t` Specifies the timeout period, in seconds.

`-v` Verbose output, on stderr.

Exit Status If the timeout occurs, then `hatimerun` exits with exit code 99 (which can be overridden to some other value using the `-e` option).

If the timeout does not occur but some other error is detected by the `hatimerun` utility (as opposed to the error being detected by the child program), then `hatimerunhatimerun` exits with exit code 98.

Otherwise, `hatimerun` exits with the child's exit status.

The `hatimerun` utility catches the signal SIGTERM. It responds to the signal by killing the child as if a timeout had occurred, and then exiting with exit code 98.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also `kill`(1), `sh`(1), `attributes`(5)

Name pmfadm – process monitor facility administration

Synopsis

```
/usr/cluster/bin/pmfadm -c nametag [-a action] [
 [-e ENV_VAR=env.var...] | -E] [-n retries] [-t period]
 [-C level#] command [args-to-command...]

/usr/cluster/bin/pmfadm -k nametag [-w timeout] [signal]

/usr/cluster/bin/pmfadm -L [-h host]

/usr/cluster/bin/pmfadm -l nametag [-h host]

/usr/cluster/bin/pmfadm -m nametag [-n retries] [-t period]

/usr/cluster/bin/pmfadm -q nametag [-h host]

/usr/cluster/bin/pmfadm -s nametag [-w timeout] [signal]
```

Description The `pmfadm` utility provides the administrative, command-line interface to the process monitor facility.

The process monitor facility provides a means of monitoring processes, and their descendents, and restarting them if they fail to remain alive. The total number of failures allowed can be specified, and limited to a specific time period. After the maximum number of failures has occurred within the specified time period, a message is logged to the console, and the process is no longer restarted.

If an *action* program has been specified, it is called when the number of failures allowed has been reached. If the *action* program exits with non-zero status, the process nametag is removed from the process monitor facility. Otherwise, the process is restarted with the original parameters passed into `pmfadm`.

Processes that are started under control of the process monitor are run as the effective user ID (`euid`) and effective group ID (`egid`) of the user that initiated the request. Only the original user, or root, can manipulate the nametag associated with those processes. Status information, however, is available to any caller, local or remote.

All spawned processes, and their descendent spawned processes, of the process that initially started are monitored. Only when the last process or sub-process exits does the process monitor attempt to restart the process.

You can run this command in the global zone or in a non-global zone. The command affects only the global or non-global zone in which you issue the command.

Options The following options are supported:

`-a` *action*
: The action program to be called when the process fails to stay alive. This program must be specified in a single argument to the `-a` option, but can be a quoted string that contains multiple components. In either case, the string is executed as specified, with two additional arguments, the event that occurred (currently only `failed`), and the nametag associated

with the process. The current directory, and PATH environment variable, are reinstantiated before the command is executed. No other environment variables are, or should be assumed to be, preserved.

If the action program exits with status `0`, the process is started over again with the original arguments that were given to `pmfadm`. Any other exit status causes the nametag to cease to exist within the scope of the process monitor.

If no `-a` option is specified, the result is the same as if there were an action script specified which always exits non-zero.

`-C` *level#*
When starting a process, monitor it and its children up to and including level *level#*. The value of *level#* must be an integer greater than or equal to zero. The original process executed is at level `0`, its children are executed at level `1`, their children are executed at level `2`, and so on. Any new fork operation produces a new level of children.

This option provides more control over which processes get monitored. It is useful for monitoring servers that fork new processes.

When this option is not specified, all children are monitored, and the original process is not restarted until it and all its children have died.

If a server forks new processes to handle client requests, it might be desirable to monitor only the server. The server needs to be restarted if it dies even if some client processes are still running. The appropriate monitoring level is `-C 0`.

If, after forking a child, the parent exits, then it is the child that needs monitoring. The level to use to monitor the child is `-C 1`. When both processes die, the server is restarted.

`-c` *nametag*
Start a process, with *nametag* as an identifier. All arguments that follow the command-line flags are executed as the process of interest. The current directory, and PATH environment variable, are reinstantiated by the process monitor facility before the command is executed. No other environment variables are, or should be assumed to be, preserved.

If *nametag* already exists, `pmfadm` exits with exit status 1, with no side effects.

I/O redirection is not supported in the command-line arguments. If this is necessary, a script should be created that performs this redirection, and used as the command that `pmfadm` executes.

`-E`
Pass the whole `pmfadm` environment to the new process. The default is not to use this option, in which case the `rpc.pmfd` environment plus the path of the `pmfadm` environment are passed.

The `-e` and `-E` options are mutually exclusive, that is, both cannot be used in the same command.

`-e ENV_VAR=`*env.value*
An environment variable in the form `ENV_VAR=`*env.value* which is passed to the execution environment of the new process. This option can be repeated, so multiple environment variables can be passed. The default is not to use this option, in which case the `rpc.pmfd` environment plus the path of the `pmfadm` environment are passed.

`-h` *host*
The name of the host to contact. Defaults to localhost.

`-k` *nametag*
Send the specified signal to the processes associated with *nametag*, including any processes associated with the action program if it is currently running. The default signal, `SIGKILL`, is sent if none is specified. If the process and its descendants exit, and there are remaining retries available, the process monitor restarts the process. The signal specified is the same set of names recognized by the `kill` command.

`-L`
Return a list of all tags running that belong to the user that issued the command, or if the user is root, all tags running on the server are shown.

`-l` *nametag*
Print out status information about *nametag*. The output from this command is useful mainly for diagnostics and might be subject to change.

`-m` *nametag*
Modify the number of retries, or time period over which to observe retries, for *nametag*. Once these parameters have been changed, the history of earlier failures is cleared.

`-n` *retries*
Number of retries allowed within the specified time period. The default value for this field is `0`, which means that the process is not restarted once it exits. The maximum value allowed is 100. A value of `-1` indicates that the number of retries is infinite.

`-q` *nametag*
Indicate whether *nametag* is registered and running under the process monitor. Returns `0` if it is, `1` if it is not. Other return values indicate an error.

`-s` *nametag*
Stop restarting the command associated with *nametag*. The signal, if specified, is sent to all processes, including the action script and its processes if they are currently executing. If a signal is not specified, none is sent. Stopping the monitoring of processes does not imply that they no longer exist. The processes remain running until they, and all of their descendents, have exited. The signal specified is the same set of names recognized by the `kill` command.

`-t` *period*
Minutes over which to count failures. The default value for this flag is `-1`, which equates to infinity. If this parameter is specified, process failures that have occurred outside of the specified period are not counted.

`-w` *timeout*

When used in conjunction with the `-s` *nametag* or `-k` *nametag* flags, wait up to the specified number of seconds for the processes associated with *nametag* to exit. If the timeout expires, `pmfadm` exits with exit status `2`. The default value for this flag is `0`, meaning that the command returns immediately without waiting for any process to exit.

If a value of `-1` is given, `pmfadm` waits indefinitely for the processes associated with the tag to exit. The `pmfadm` process does not release the RPC server thread that it uses until the RPC timeout period is reached. Therefore, avoid setting the `-w` *timeout* value to -1 unnecessarily.

Examples

EXAMPLE 1 Starting a Sleep Process That Will Not be Restarted

The following example starts a sleep process named `sleep.once` that will not be restarted once it exits:

```
example% pmfadm -c sleep.once /bin/sleep 5
```

EXAMPLE 2 Starting a Sleep Process and Restarting It

The following example starts a sleep process and restarts it, at most, one time:

```
example% pmfadm -c sleep.twice –n 1 /bin/sleep 5
```

EXAMPLE 3 Starting a Sleep Process and Restarting It

The following examples start a sleep process and restarts it, at most, twice per minute. It calls `/bin/true` when it fails to remain running beyond the acceptable number of failures:

```
example% pmfadm -c sleep.forever –n 2 –t 1 –a /bin/true /bin/sleep 60
```

EXAMPLE 4 Listing the Current Status of the `sleep.forever` Nametag

The following command lists the current status of the `sleep.forever` nametag:

```
example% pmfadm -l sleep.forever
```

EXAMPLE 5 Sending a `SIGHUP` to All Processes

The following command sends a `SIGHUP` to all processes associated with `sleep.forever`, waiting up to five seconds for all processes to exit.

```
example% pmfadm -w 5 -k sleep.forever HUP
```

EXAMPLE 6 Stopping the Monitoring of Processes and Sending a SIGHUP

The following command stops monitoring (restarting) processes associated with `sleep.forever`, and sends a `SIGHUP` to any processes related to it. This command returns as soon as the signals have been delivered, but possibly before all processes have exited.

```
example% pmfadm -s sleep.forever HUP
```

EXAMPLE 7 Listing All Tags Running That Belong to the User

If a user issues the following commands:

```
example% pmfadm -c sleep.once /bin/sleep 30
example% pmfadm -c sleep.twice /bin/sleep 60
example% pmfadm -c sleep.forever /bin/sleep 90
```

the output of the following command:

```
example% pmfadm -L
```

is

```
sleep.once sleep.twice sleep.forever
```

Exit Status

The following exit values are returned:

`0`	Successful completion.
`1`	*nametag* doesn't exist, or there was an attempt to create a nametag that already exists.
`2`	The command timed out.
other nonzero	An error occurred.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also

`kill`(1), `truss`(1), rpc.pmfd(1M), `attributes`(5)

Notes

To avoid collisions with other controlling processes. The `truss` command does not allow tracing a process that it detects as being controlled by another process by way of the `/proc` interface. Because the `rpc.pmfd` daemon prior to Solaris 10 OS uses the `/proc` interface to monitor processes and their descendants, those processes that are submitted to `rpc.pmfd` by way of the `pmfadm` command cannot be traced or debugged. As of the Solaris 10 OS release, this restriction no longer applies.

Name pnmd – Public Network Management (PNM) service daemon

Synopsis

```
/usr/cluster/bin/pnmd [-d [-t [tracefile]]]
```

Description `pnmd` is a server daemon for the Public Network Management (PNM) module. It is usually started up at system boot time. When it is started, it starts the PNM service.

The `in.mpathd`(1M) daemon does adapter testing and intra-node failover for all IP Network Multipathing (IPMP) groups in the local host.

`pnmd` keeps track of the local host's IPMP state and facilitates inter-node failover for all IPMP groups.

You can use this command only in the global zone.

Options The following options are supported:

`-d`	Display debug messages on `stderr`.
`-t` *tracefile*	When used with the `-d` option, it causes all debug messages to be redirected to *tracefile*. If *tracefile*. is omitted, `/var/cluster/run/pnmd.log` is used.

Diagnostics `pnmd` is a daemon and has no direct `stdin`, `stdout`, or `stderr` connection to the outside. All diagnostic messages are logged through `syslog`(3C).

Notes `pnmd` must be run in superuser mode.

Due to the volume of debug messages generated, do not use the `-t` option for an extended period of time.

`pnmd` is started by the `pnm` startup script. It is started under the Process Monitoring Facility daemon `pmfd`. As such, if `pnmd` is killed by a signal, it is automatically restarted by `pmfd`.

The `SIGTERM` signal can be used to kill `pnmd` gracefully. Other signals should not be used to kill the daemon.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also `ifconfig`(1M), `in.mpathd`(1M), `syslog`(3C), `attributes`(5)

Name rdt_setmtu – set the MTU size in RSMRDT driver

Synopsis

```
/usr/cluster/bin/rdt_setmtu [MTU size]
```

Description The `rdt_setmtu` command takes number of bytes as new MTU size and sets the global MTU size in RSMRDT driver. The RSMRDT driver uses the new MTU size for all the new instantiations of RSM connections. The existing RSM connections continue to use the old MTU size value. The MTU size should be a multiple of `64` (`0x40`) bytes otherwise `rdt_setmtu` does not set the MTU size in RSMRDT driver and returns an error. The `rdt_setmtu` when running without any argument, displays the MTU size of RSMRDT driver.

You can use this command only in the global zone.

Operands The following operands are supported:

`MTU` *size* MTU size in bytes.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred while setting MTU size.

This utility writes an error message to `stderr` when it exits with non-zero status.

Attributes See `attributes`(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscrdt
Interface Stability	Evolving

See Also `attributes`(5)

Name rpc.pmfd, pmfd – RPC-based process monitor server

Synopsis

```
/usr/cluster/lib/sc/rpc.pmfd
```

Description The `rpc.pmfd` daemon is the Sun RPC server for serving the process monitor facility that is used by Sun Cluster software. This daemon initially starts when the system comes up.

The `rpc.pmfd` daemon must be started as superuser so commands that are queued to be monitored can be run as the user that submitted them.

You can run this command in the global zone or in a non-global zone. The command affects only the global or non-global zone in which you issue the command.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also `truss`(1)`attributes`(5)

Diagnostics Diagnostic messages are normally logged to the console.

Notes To avoid collisions with other controlling processes, the `truss` command does not allow tracing a process that it detects as being controlled by another process by way of the `/proc` interface. Because the `rpc.pmfd` daemon prior to the Solaris 10 OS uses the `/proc` interface to monitor processes and their descendants, those processes that are submitted to the `rpc.pmfd` daemon by way of the `pmfadm` command cannot be traced or debugged. As of the Solaris 10 OS release, this restriction no longer applies.

Name sccheck – check for and report on vulnerable Sun Cluster configurations

Synopsis

```
sccheck [-b] [-h nodename[,nodename]...] [-o output-dir]
 [-s severity] [-v verbosity]

sccheck [-b] [-W] [-h nodename[,nodename]...] [-o output-dir] [-v verbosity]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `sccheck` utility examines Sun Cluster nodes for known vulnerabilities and configuration problems, and it delivers reports that describe all failed checks, if any. The utility runs one of these two sets of checks, depending on the state of the node that issues the command:

- **Preinstallation checks –** When issued from a node that is not running as an active cluster member, the `sccheck` utility runs preinstallation checks on that node. These checks ensure that the node meets the minimum requirements to be successfully configured with Sun Cluster software.
- **Cluster configuration checks –** When issued from an active member of a running cluster, the `sccheck` utility runs configuration checks on the specified or default set of nodes. These checks ensure that the cluster meets the basic configuration required for a cluster to be functional. The `sccheck` utility produces the same results for this set of checks regardless of which cluster node issues the command.

The `sccheck` utility runs configuration checks and uses the `explorer`(1M) utility to gather system data for check processing. The `sccheck` utility first runs single-node checks on each *nodename* specified, then runs multiple-node checks on the specified or default set of nodes.

Each configuration check produces a set of reports that are saved in the specified or default output directory. For each specified *nodename*, the `sccheck` utility produces a report of any single-node checks that failed on that node. Then the node from which `sccheck` was run produces an additional report for the multiple-node checks. Each report contains a summary that shows the total number of checks executed and the number of failures, grouped by check severity level.

Each report is produced in both ordinary text and in XML. The DTD for the XML format is available in the `/usr/cluster/lib/sccheck/checkresults.dtd` file. The reports are produced in English only.

The `sccheck` utility is a client-server program in which the server is started when needed by the `inetd` daemon. Environment variables in the user's shell are not available to this server. Also, some environment variables, in particular those that specify the non-default locations of Java and Sun Explorer software, can be overridden by entries in the `/etc/default/sccheck`

file. The ports used by the `sccheck` utility can also be overridden by entries in this file, as can the setting for required minimum available disk space. The server logs error messages to `syslog` and the console.

You can use this command only in the global zone.

Options The following options are supported:

`-b`
Specifies a brief report. This report contains only the summary of the problem and the severity level. Analysis and recommendations are omitted.

You can use this option only in the global zone.

You need `solaris.cluster.system.read` RBAC authorization to use this command option. See `rbac`(5).

`-h` *nodename*[,*nodename*]…
Specifies the nodes on which to run checks. If the `-h` option is not specified, the `sccheck` utility reports on all active cluster members.

You can use this option only in the global zone.

This option is only legal when issued from an active cluster member.

`-o` *output-dir*
Specifies the directory in which to save reports.

You can use this option only in the global zone.

The *output-dir* must already exist or be able to be created by the `sccheck` utility. Any previous reports in *output-dir* are overwritten by the new reports.

If the `-o` option is not specified, `/var/cluster/sccheck/reports.`*yyyy–mm–dd*:*hh*:*mm*:*ss* is used as *output-dir* by default, where *yyyy–mm–dd*:*hh*:*mm*:*ss* is the year-month-day:hour:minute:second when the directory was created.

`-s` *severity*
Specifies the minimum severity level to report on.

You can use this option only in the global zone.

The value of *severity* is a number in the range of 1 to 4 that indicates one of the following severity levels:

1. Low
2. Medium
3. High
4. Critical

Each check has an assigned severity level. Specifying a severity level will exclude any failed checks of lesser severity levels from the report. When the `-s` option is not specified, the default severity level is 0, which means that failed checks of all severity levels are reported.

The `-s` option is mutually exclusive with the `-W` option.

`-v` *verbosity*
Specifies the `sccheck` utility's level of verbosity.

You can use this option only in the global zone.

The value of *verbosity* is a number in the range of 0 to 2 that indicates one of the following verbosity levels:

- 0: No progress messages. This level is the default.
- 1: Issues `sccheck` progress messages.
- 2: Issues Sun Explorer and more detailed `sccheck` progress messages.

You need `solaris.cluster.system.read` RBAC authorization to use this command option. See `rbac`(5).

The `-v` option has no effect on report contents.

`-W`
Disables any warnings. The report generated is equivalent to `-s 3`.

You can use this option only in the global zone.

The `-W` option is mutually exclusive with the `-s` option. The `-W` option is retained for compatibility with prior versions of the `sccheck` utility.

You need `solaris.cluster.system.read` RBAC authorization to use this command option. See `rbac`(5).

Exit Status The following exit values are returned:

`0`	The command completed successfully. No violations were reported.
`1-4`	The code indicates that the highest severity level of all violations was reported.
`100+`	An error has occurred. Some reports might have been generated.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu, SUNWscsck
Interface Stability	Evolving

Files `/etc/default/sccheck`

`/usr/cluster/lib/sccheck/checkresults.dtd`

`/var/cluster/sccheck/reports.`*yyyy–mm–dd*`:`*hh*`:`*mm*`:`*ss*

See Also Intro(1CL), `explorer`(1M), sccheckd(1M), scinstall(1M), `attributes`(5)

Sun Cluster Software Installation Guide for Solaris OS, *Sun Cluster System Administration Guide for Solaris OS*

Name sccheckd – service for the cluster configuration check utility

Synopsis

```
sccheckd
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `sccheckd` service is the server side of the client-server cluster configuration check utility. This utility is called by the `sccheck` command.

The `inetd` daemon starts the cluster configuration check service. The service reads the `/etc/default/sccheck` file at startup and during execution. The service logs diagnostics and error messages to `syslog` and the console. The `sccheckd` service has no direct connection to `stdin`, `stdout`, or `stderr`.

The `sccheckd` service exits when the last client connection exits.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

Files `/etc/default/sccheck`

See Also Intro(1CL), sccheck(1M), `inetd`(1M)

Name scconf – update the Sun Cluster software configuration

Synopsis

```
scconf -a [-Hv] [-h node_options] [-A adapter_options]
 [-B switch_options] [-m cable_options] [-P privatehostname_options]
 [-q quorum_options] [-D devicegroup_options] [-T authentication_options]

scconf -c [-Hv] [-C cluster_options] [-A adapter_options]
 [-B switch_options] [-m cable_options] [-P privatehostname_options]
 [-q quorum_options] [-D devicegroup_options] [-S slm_options]
 [-T authentication_options] [-w heartbeat_options]

scconf -r [-Hv] [-h node_options] [-A adapter_options]
 [-B switch_options] [-m cable_options] [-P privatehostname_options]
 [-q quorum_options] [-D devicegroup_options] [-T authentication_options]

scconf -p [-Hv [v]]

scconf [-H]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scconf` command manages the Sun Cluster software configuration. You can use `scconf` to add items to the configuration, to change properties of previously configured items, and to remove items from the configuration. In each of these three forms of the command, options are processed in the order in which they are typed on the command line. All updates associated with each option must complete successfully before the next option is considered.

The `scconf` command can also be used to register VxVM disk groups, SVM metasets, and raw disk groups when the disk groups or metasets consist of disks that use controller-based replication for data availability. Hitachi TrueCopy is an example of controller-based replication. Before using the `scconf` command to register disk groups and metasets, ensure that all disks in the disk group are either replicated or non-replicated, but not both. Also, you must execute the `scdidadm` command with the `-T` or `-t` options or the `cldevice replicate` command. These commands configure the DID device to use controller-based replication. For more information, see the scdidadm(1M) man page or the cldevice(1CL) man page.

The `scconf` command can only be run from an active cluster node. As long as the node is active in the cluster, it makes no difference which node is used to run the command. The results of running the command are always the same, regardless of the node used.

The `-p` option of `scconf` enables you to print a listing of the current configuration.

All forms of the `scconf` command accept the `-H` option. Specifying `-H` displays help information, and all other options are ignored and not executed. Help information is also printed when `scconf` is invoked without options.

You can use this command only in the global zone.

Options

Basic Options

The following option is common to all forms of the `scconf` command:

`-H`
: If this option is specified on the command line at any position, it prints help information. All other options are ignored and are not executed. Help information is also printed if `scconf` is invoked with no options.

  You can use this option only in the global zone.

The following options modify the basic form and function of the `scconf` command. None of these options can be combined on the same command line.

`-a`
: Specifies the `add` form of the `scconf` command.

  You can use this option only in the global zone.

  The `-a` option can be used to add or initialize most of the items that are used to define the software configuration of a Sun Cluster. Additional options are used with `-a` to specify elements (adapter, switch, or device group options, for example) and their associated properties to be added. Any number of these additional options can be combined on the same command line, as long as they are for use with the `-a` option.

`-c`
: Specifies the `change` form of the `scconf` command.

  You can use this option only in the global zone.

  The `-c` option is used to change properties of items already configured as part of the Sun Cluster software configuration. Additional options are used with `-c` to specify new or changed properties. Any number of these additional options can be combined on the same command line, as long as they are for use with the `-c` option.

`-p`
: Specifies the `print` form of the `scconf` command.

  You can use this option only in the global zone.

  The `-p` option prints a listing of the current Sun Cluster configuration elements and their associated properties that you can configure with `scconf`. This option can be combined with one or more `-v` options to print more verbose listings.

`-r`
: Specifies the `remove` form of the `scconf` command.

  You can use this option only in the global zone.

The `-r` option is used to remove items from the Sun Cluster software configuration. Additional options are used with `-r` to specify the items to delete from the configuration. Any number of these additional options can be combined on the same command line, as long as they are for use with the `-r` option.

Additional Options

The following additional options can be combined with one or more of the previously described basic options. Refer to the SYNOPSIS section to see the options that can be used with each form of `scconf`.

The additional options are as follows:

`-A` *adapter_options*

Adds, removes, or changes the properties of a cluster transport adapter. The node on which the given adapter is hosted need not be active in the cluster for these operations to succeed. The `-A` *adapter_options* for each of the three forms of the command that accept `-A` are described here.

- Use this syntax to specify `-A` *adapter_options* for the `add` form of the command:

  `–A name=`*adaptername*`,node=`*node*`[,vlanid=`*vlanid*`][,state=`*state*`] \`
  `[,`*other_options*`]`

- Use this syntax to specify `-A` *adapter_options* for the `change` form of the command:

  `–A name=`*adaptername*`,node=`*node*`[,state=`*state*`] \`
  `[,`*other_options*`]`

- Use this syntax to specify `-A` *adapter_options* for the `remove` form of the command:

  `–A name=`*name*`,node=`*node*

The `-A` option supports the following suboptions:

`name=`*adaptername*

Specifies the name of an adapter on a particular node. This suboption must be included with each occurrence of the `-A` option.

adaptername is constructed from a *device name*, immediately followed by a *physical-unit* number (for example, `hme0`).

`node=`*node*

Specifies the name of an adapter on a particular node. A `node` suboption is required for each occurrence of the `-A` option.

The *node* can be given either as a node name or node ID.

`state=`*state*

Changes the state of the adapter. You can use this suboption with the `change` form of the command. The `state` can be set to either `enabled` or `disabled`.

When an adapter is added to the configuration, its state is always set to `disabled`. By default, adding a cable to any of the ports on an adapter changes the state of both the port and the adapter to `enabled`. See `-m` *cable_options*.

Disabling an adapter also has the effect of disabling all ports associated with that adapter. However, enabling an adapter does not result in the enabling of its ports. To enable an adapter port, you must enable the cable to which the port is connected.

`trtype=`*type*
Specifies the transport type. This suboption must be included when `-A` is used with the `add` form of the command.

An example of a transport *type* is `dlpi`. See sctransp_dlpi(7p).

[`vlanid=`*vlanid*]
Specifies the VLAN ID of the tagged-VLAN adapter.

[*other_options*]
If other options are available for a particular adapter type, they can be used with `-A` in the add and change forms of the command. Refer to the cluster transport adapter man pages (for example, scconf_transp_adap_hme(1M), scconf_transp_adap_eri(1M), and scconf_transp_adap_sci(1M)) for information about special options.

You need `solaris.cluster.transport.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-B` *switch_options*
Adds, removes, or changes the properties of a cluster transport switch, also called transport junction.

Examples of such devices can include, but are not limited to, Ethernet hubs, other switches of various types, and rings.

The `-B` *switch_options* for each of the three forms of the command that accept `-B` are described here.

- Use this syntax to specify `-B` *switch_options* for the `add` form of the command:

  –`B type=`*type*`,name=`*name*`[,`*other_options*`]`
- Use this syntax to specify `-B` *switch_options* for the `change` form of the command:

  –`B name=`*name*`[,state=`*state*`][,`*other_options*`]`
- Use this syntax to specify `-B` *switch_options* for the `remove` form of the command:

  –`B name=`*name*

The `-B` option supports the following suboptions:

`name=`*name*
Specifies the name of a cluster transport switch. A `name` suboption must be included with each occurrence of the `-B` option.

name can be up to 256 characters in length. It is made up of either letters or digits, with the first character being a letter. Each transport switch name must be unique across the namespace of the cluster.

`state=`*state*
Changes the state of a cluster transport switch. This suboption can be used with a `-B change` command. `state` can be set to either `enabled` or `disabled`.

When a switch is added to the configuration, its state is always set to `disabled`. By default, adding a cable to any of the ports on a switch changes the state of both the port and the switch to `enabled`. See `-m` *cable_options*.

Disabling a switch also has the effect of disabling all ports associated with that switch. However, enabling a switch does not result in the enabling of its ports. To enable a switch port, you must enable the cable to which the port is connected.

`type=`*type*
Specifies a cluster transport switch type. This suboption must be included when `-B` is used with the `add` form of the command.

Ethernet hubs and SCI switches are examples of cluster transport switches of type `switch`. The man pages scconf_transp_jct_dolphinswitch(1M) and scconf_transp_jct_etherswitch(1M) contain more information.

[*other_options*]
When other options are available for a particular switch type, they can be used with `-B` in the `add` and `change` forms of the command. Refer to the cluster transport switch man pages (for example, scconf_transp_jct_dolphinswitch(1M) and scconf_transp_jct_etherswitch(1M)) for information about special options.

You need `solaris.cluster.transport.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-C` *cluster_options*
Changes the name of the cluster itself. This option can only be used with the `change` form of the command.

Specify *cluster_options* for the `change` form of the command as follows:

`-C cluster=`*clustername*

This form of the command changes the name of the cluster to *clustername*.

`-D` *devicegroup_options*
Adds device groups to the configuration, changes or resets properties of existing device groups, or removes groups from the Sun Cluster device groups configuration. Other device group options (*other_options*) play a crucial role in adding or changing device groups and their options. Pay special attention to the man pages for the type-dependent device group options (for example, scconf_dg_vxvm(1M), scconf_dg_svm(1M), and scconf_dg_rawdisk(1M)) when configuring any device group. Not all device group types support all three forms of the `-D` option. For example, `svm` device groups can normally only be used with the `change` form of the command to change certain attributes, such as the ordering of the node preference list.

The add form of the command can be used to either create device groups or to add nodes to existing device groups. For some device group types, the add form can also be used to add devices to a group. The change form of the command registers updates to change certain attributes associated with a group. The remove form of the command is used to either remove an entire device group or one or more of a group's components.

The -D *devicegroup_options* for each of the three forms of the scconf command that accept -D are as follows:

Add:

```
-D type=type,name=name[,nodelist=node[:node]...]
        [,preferenced={true | false}]
        [,numsecondaries=integer]
        [,failback={enabled | disabled}][,other_options]
```

Change:

```
-D name=name[,nodelist=node[:node]...]
        [,preferenced={true | false}]
        [,numsecondaries=integer]
        [,failback={enabled | disabled}][,other_options]
```

Remove:

```
-D name=name,nodelist=node[:node]...
```

The -D option supports the following suboptions:

name=*name*
: The name of the device group. This name must be supplied with all three forms of the command.

nodelist=*node*[:*node*]…
: A list of potential primary nodes that is required for some device group types when adding a group to the cluster. For the vxvm device group type, the concept of primary nodes does not apply when the localonly property is set to true. Refer to the man pages for the type-dependent device group for more information.

  The nodelist suboption is required when you set the preferenced suboption to true.

  With the add form of the command, the nodelist is, by default, an ordered list indicating the preferred order in which nodes should attempt to take over as the primary node for a device group. However, if the preferenced suboption is set to false (see the next subsection), the first node to access a device in the group automatically becomes the primary node for that group. The preferenced suboption cannot be used when adding nodes to an existing device group. However, the preferenced suboption can be used when you create the group for the first time, or with the change form of the command.

To change the primary node order preference, you must specify the complete list of cluster nodes in the `nodelist` in the order that you prefer. You must also set the `preferenced` suboption to `true`.

When used with the `remove` form of the command, the `nodelist` suboption is used to remove the indicated nodes from the device group. Only by not providing a `nodelist` can the entire device group be removed. Simply removing all of the nodes from a device group does not necessarily remove that group.

`type=`*type*
The type of device group. The type must be used with the `add` form of the command to indicate the type of device group to create (for example, `vxvm` or `rawdisk`).

[`failback=`{`enabled` | `disabled`}]
Enables or disables the `failback` behavior of a device group with either the `add` or the `change` form of the command.

Specifies the behavior of the system should a device group primary node leave the cluster membership and later return.

When the node leaves the cluster membership, the device group fails over to the secondary node. When the failed node rejoins the cluster membership, the device group can either continue to be mastered by the secondary node, or fail back to the original primary node.

If `failback` is `enabled`, the device group becomes mastered by the original primary node. If `failback` is `disabled`, the device group continues to be mastered by the secondary node.

By default, `failback` is `disabled`.

[`numsecondaries=`*integer*]
Enables you to dynamically change the desired number of secondary nodes for a device group. A device group is an HA service that requires one node to act as a primary node and one or more nodes to act as secondary nodes. The secondary nodes of a device group are able to take over and act as the primary node if the current primary node fails.

This integer value should be greater than 0 but less than the total number of nodes in the specified group. The default is 1.

A system administrator can use the `numsecondaries` suboption to change the number of secondary nodes for a device group while maintaining a given level of availability. If a node in a device group is removed from the secondary nodes list, it is not able to take over and act as a primary node until it is converted back to a secondary node. Before making a change to the number of secondary nodes, you need to assess the impact on the secondary global file system.

The `numsecondaries` suboption only applies to nodes in a device group that are currently in cluster mode and can be used together with the node's `preferenced` suboption. If a device's `preferenced` suboption is enabled, the nodes that are least preferred are removed from the secondary nodes list first. If no node in a device group is flagged as preferred, the cluster randomly picks the node to remove.

When a device group's actual number of secondary nodes drops to less that the desired level due to node failures, nodes that were removed from the secondary nodes list are added back to the secondary list of nodes if they are currently in a cluster, belong to the device group, and are not currently a primary or a secondary node. The conversion starts with the node in the device group with the highest preference until the number of desired secondary nodes is matched.

If a node in the device group has a higher preference than an existing secondary node and joins the cluster, the node with the least preference is removed from the secondary nodes list and is replaced by the newly added node. This replacement only occurs when there are more actual secondary nodes than the desired level.

To set the desired number of secondary nodes to the system default (without having to know the default value), issue one of these commands:

```
# scconf -aD type=vxvm,name=foo, \
nodelist=node1:node2,numsecondaries=
```

or

```
# scconf -cD name=foo,numsecondaries=
```

The `numsecondaries` suboption can only be used with the `-a` option when a device group is created. The `numsecondaries` suboption cannot be used with the `-a` option to add a host to an existing device group.

[`preferenced={true | false}`]
Indicates the status of the preferred order of potential primary nodes for a device group. As long as the `preferenced` suboption is not set to `false`, node lists for newly created device groups indicate a preferred order in which nodes attempt to take over as the primary node for a device group.

If you set the `preferenced` suboption to `true`, you must also use the `nodelist` subopton to specify the entire node list.

If the `preferenced` suboption is not specified with an `add` that is used to create a device group, it is, by default, `false`. However, if the preferenced suboption is not specified with a `change`, it is, by default, set to `true` when `nodelist` is given.

The `preferenced` suboption cannot be used with an `add` that is used to add nodes to an established device group. In this case, the established node preference list setting is used.

[*other_options*]
You can use other device group type-dependent options with either the `add` or `change` form of the command. Refer to the appropriate man pages for more information (for example, scconf_dg_vxvm(1M), scconf_dg_svm(1M), and scconf_dg_rawdisk(1M)).

You need `solaris.cluster.device.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-h` *node_options*
Adds or removes a node from the cluster configuration database. When used with the `add` form of `scconf`, both the new name and an internally generated node ID are added to the cluster configuration database. In addition, the new node is given a disk reservation key and a quorum vote count of zero. The name that is assigned to access the node over the cluster interconnect is initialized to `clusternode`*nodeid*`-priv`. See the `-p` option to learn more about printing configuration elements and their associated properties.

`scconf` cannot be used by itself to add a new node to the cluster. You can only use `scconf` to update the configuration database itself. `scconf` does not copy the configuration database onto the new node or create the necessary node identifier on the new node. To add a node to a cluster, use scinstall(1M).

When used with the `remove` form of `scconf`, all references to the node, including the last transport cable, all resource group references, and all device group references must be removed before `scconf` can be used to completely remove the node from the cluster configuration.

The node to be removed must not be configured for any quorum devices. In addition, you cannot remove a node from a three-node cluster unless there is at least one shared quorum device configured.

The system administration procedures in the Sun Cluster documentation describe how to remove a cluster node in more detail.

You must specify the `node=`*node* suboption with any occurrence of the `-h` option. For the `add` form of the command, the given *node* must be a node name.

Use this syntax to specify the `-h` *node_options* for the `add` form of the command:

`–h node=`*nodename*

For the `remove` form of the command, the *node* can be given either as a node name or node ID. Use this syntax to specify the `-h` *node_options* for the `remove` form of the command:

`–h node=`*node*

You need `solaris.cluster.node.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-m` *cable_options*

Helps to establish the cluster interconnect topology. This option helps by configuring the cables that are connecting the various ports that are found on the cluster transport adapters and switches. Each new cable typically maps a connection either between two cluster transport adapters or between an adapter and a port on a transport switch. The `-m` *cable_options* for each of the forms of the command that accept `-m` are as follows:

- Use this syntax to specify the `-m` *cable_options* for the `add` form of the command:

  `–m endpoint=[`*node*`:]`*name*`[@`*port*`],`
  `endpoint=[`*node*`:]`*name*`[@`*port*`][,noenable]`

- Use this syntax to specify the `-m` *cable_options* for the `change` form of the command:

  `–m endpoint=[`*node*`:]`*name*`[@`*port*`],state=`*state*

- Use this syntax to specify the `-m` *cable_options* for the `remove` form of the command:

  `–m endpoint=[`*node*`:]`*name*`[@`*port*`]`

The `-m` option supports the following suboptions:

`endpoint=[`*node*`:]`*name*`[@`*port*`]`

Must be included with each occurrence of the `-m` option. For the `add` form of the command, two `endpoint` options must be specified. The *name* component of the option argument is used to specify the name of either a cluster transport adapter or cluster transport switch at one of the endpoints of a cable. If a *node* component is given, the *name* is the name of a cluster transport adapter. Otherwise, the *name* is the name of a cluster transport switch.

If a *port* component is not given, an attempt is made to assume a default port name. The default port for an adapter is always `0`. The default port name for a switch endpoint is equal to the node ID of the node attached to the other end of the cable. Refer to the cluster transport adapter and cluster transport switch man pages for more information about *port* assignments and other requirements (for example,scconf_transp_adap_hme(1M), scconf_transp_adap_eri(1M), scconf_transp_adap_sci(1M), scconf_transp_jct_etherswitch(1M), and scconf_transp_jct_dolphinswitch(1M)). Before a cable can be added, the adapters and switches at each of the two endpoints of the cable must already be configured (see `-A` and `-B`).

`noenable`

Can be used when adding a cable to the configuration. By default, when you add a cable, the state of the cable, the two ports to which it is connected, and the adapters or switches on which the ports are found, are set to `enable`. But, if `noenable` is specified when you add a cable, the cable and its two endpoints are added in the disabled state. The state of the adapters or switches on which the ports are found remains unchanged.

`state=`*state*
Changes the state of a cable and the two endpoints to which it is connected. When a cable is enabled, the cable, its two ports, and the adapters or switches that are associated with those two ports are all enabled. However, when a cable is disabled, only the cable and its two ports are disabled. The state of the adapters or switches that are associated with the two ports remains unchanged. By default, the state of a cable and its endpoints is always set to `enabled` at the time that the cable is added to the configuration. But to add a cable in the `disabled` state, use the `noenable` suboption as part of an add operation.

You need `solaris.cluster.transport.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-P` *privatehostname_options*
For a node, adds or changes the private hostname. For a non-global zone, adds or removes the private IP address and the private hostname, or changes the private hostname.

When used with the `add` (`-a`) form of the command, the `-P` option specifies one of the following actions:

- When a node is specified, the command assigns the specified hostname alias to use for IP access of the specified node over the private cluster interconnect, or transport. If not otherwise assigned or if reset, the default private hostname for the node is `clusternode`*nodeid*`-priv`.
- When a non-global zone is specified, the command assigns the specified zone private IP address and plumbs this IP address over the private interconnect. This IP address is automatically chosen from available addresses in the cluster private IP address range.

  The command also assigns the specified hostname alias to use for IP access of the specified zone over the private cluster interconnect. The hostname must not be used by any other zone or node in the enterprise.

  The private IP address range that is configured for the cluster must support the increased number of private IP addresses that are used in the cluster. Ensure that the private IP address range can support the added private IP address before you assign one to a zone. See the scprivipadm(1M) man page for more information.

When used with the `change` (`-c`) form of the command, the `-P` option changes the hostname alias for the specified node or non-global zone.

When used with the `remove` (`-r`) form of the command, the `-P` option frees the IP address that is assigned to the specified non-global zone. This removal makes the IP address available for use elsewhere. The zone private hostname is not removed, but it will no longer be resolved to the removed private IP address. The `remove` form of the command is only valid for zones.

Private hostnames should never be stored in the `hosts`(4) database. A special `nsswitch` facility (see `nsswitch.conf`(4)) performs all hostname lookups for private hostnames.

The *privatehostname_options* for each of the forms of the command that accept `-P` are as follows:

Add:

```
–P node=node[,privatehostname=hostalias]
–P node=node:zone,zprivatehostname=hostalias
```

Change:

```
–P node=node[,privatehostname=hostalias]
–P node=node:zone,zprivatehostname=hostalias
```

Remove:

```
-P node=node:zone
```

The `-P` option supports the following suboptions:

`node=`*node*
: Provides the name or ID of the node to be assigned the specified private hostname, or host alias, that is supplied with the `privatehostname` suboption.

`node=`*node*`:`*zone*
: Provides the name of the non-global zone to be assigned a zone private IP address and the specified private hostname, or host alias, that is supplied with the `zprivatehostname` suboption.

`privatehostname=`*hostalias*
: Supplies the host alias to be used for accessing the node over the private cluster interconnect, or transport. If no `privatehostname` suboption is specified, the private hostname for the specified `node` is reset to the default.

`zprivatehostname=`*hostalias*
: Supplies the host alias to be used for accessing the non-global zone over the private cluster interconnect, or transport. You must specify the `zprivatehostname` suboption; there is no default host alias for zones.

You need `solaris.cluster.transport.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-q` *quorum_options*
: Manages shared cluster quorum devices and various cluster quorum properties. Pay special attention to the man pages for type-dependent quorum device options (for example, scconf_quorum_dev_scsi(1M) and scconf_quorum_dev_netapp_nas(1M).

  Caution – Devices that use controller-based replication cannot be used as quorum devices in the Sun Cluster environment. If you specify a device that uses controller-based replication using the `-q` option, the `scconf` command returns an error.

The `add` and `remove` forms of the command add and remove shared quorum devices to or from the configuration. The `change` form of the command changes various cluster quorum configuration properties or states. The `-q` *quorum_options* available for each of the three forms of the command can be used to change the cluster quorum configuration as follows:

Add:

```
–q name=devicename,type={scsi | netapp_nas}
```

For SCSI quorum devices only:

```
-q autoconfig[,noop]
```

Change:

```
-q node=node,{maintstate | reset}
–q name=devicename,{maintstate | reset}
–q reset
-q installmode
```

For SCSI quorum devices only:

```
-q autoconfig[,noop]
```

Remove:

```
–q name=devicename
```

When `scconf` is interrupted or fails while performing quorum-related operations, quorum configuration information can become inconsistent in the cluster configuration database. If this occurs, either run the same `scconf` command again or run it with the `reset` suboption to reset the quorum information.

The `-q` option supports the following suboptions:

`autoconfig`
When used with the `add` form of the command, automatically chooses and assigns one quorum device in the two-node cluster. The quorum device is chosen from the available devices. If a quorum device is already configured, the command aborts.

When used with the `change` form of the command, automatically chooses and assigns one device that replaces all existing quorum devices in the two-node cluster. The quorum device is chosen from the available devices.

All available devices in the cluster must be qualified to be a quorum device. The `autoconfig` suboption does not assess whether an available device is qualified to be a quorum device.

If the cluster contains more than two nodes, the `autoconfig` suboption makes no changes to the quorum configuration. Do not use the `autoconfig` suboption if you intend to configure a NAS device as quorum.

`installmode`
Forces the cluster back into installation mode. While in `installmode`, nodes do not attempt to reset their quorum configurations at boot time. Also, while in this mode, many administrative functions are blocked. When a cluster is first installed, it is set up with `installmode` set. Once all of the nodes have joined the cluster for the first time, and shared quorum devices have been added to the configuration, issue `scconf -c -q reset` to reset the vote counts to their default values and to clear the `installmode` setting.

`name=`*devicename*
Specifies the name of an attached shared storage device to use when adding or removing a shared quorum device to or from the cluster. This suboption can also be used with the `change` form of the command to change the state of a quorum device.

Each quorum device must be connected, or ported, to at least two nodes in the cluster. It is not possible to use a non-shared disk as a quorum device.

The `change` form of `scconf` can be used with `-q name` to either put the device into a maintenance state or to reset the device's quorum configuration to the default. While in maintenance state, the device takes on a vote count of zero and, so, does not participate in forming quorum. When reset to the default, the vote count for the device is changed to N-1, where N is the number of nodes with nonzero vote counts that have ports to the device.

`node=`*node*
When used with the `add` form of the command, selects the nodes that should be configured with ports to the shared quorum device being added. This suboption can also be used with the `change` form of the command to change the quorum state of a node.

When the `node` suboption is used with the `change` form of the quorum update command, it is used to either place a node into maintenance state or to reset the node's quorum configuration to the default.

You must shut down a node before you can put it into maintenance state. `scconf` returns an error if you attempt to put a cluster member into maintenance state.

While in maintenance state, the node takes on a vote count of zero and, so, does not participate in quorum formation. In addition, any shared quorum devices configured with ports to the node have their vote counts adjusted down by one to reflect the new state of the node. When the node is reset to the default, its vote count is reset to 1 and the shared quorum device vote counts are re-adjusted back up. Unless the cluster is in `installmode`, the quorum configuration for each node is automatically reset at boot time.

A *node* can be specified as either a node name or a node ID.

`type=`*type*
When used with the `add` form of the command, specifies the type of quorum device to create.

`scsi`
Specifies a shared disk quorum device. See scconf_quorum_dev_scsi(1M) for SCSI-type-specific options.

`netapp_nas`
Specifies a Network Appliance NAS quorum device. See scconf_quorum_dev_netapp_nas(1M) for NAS-type-specific options.

{`maintstate`}
When used as a flag with the `change` form of the command, for either the `globaldev` or `node` suboptions, puts a shared quorum device or node into a quorum maintenance state. When in maintenance state, a shared device or node no longer participates in quorum formation. This feature can be useful when a node or device must be shut down for an extended period of maintenance. Once a node boots back into the cluster, under usual circumstances, it removes itself from maintenance mode.

It is not legal to specify both `maintstate` and `reset` with the same `-q` option.

[`,noop`]
Is valid with the `autoconfig` suboption. The command prints to standard output the list of quorum devices that the `autoconfig` suboption would add or change. The `autoconfig,noop` suboption makes no changes to the quorum configuration.

{`reset`}
When used as a flag with the `change` form of the command, resets the configured quorum vote count of a shared quorum device or node. This option can be combined with either the `globaldev` or `node` suboptions, or it can be its own suboption.

If used by itself, the entire quorum configuration is reset to the default vote count settings. In addition, if `installmode` is set, it is cleared by a global quorum configuration reset. `installmode` cannot be reset on a two-node cluster unless at least one shared quorum device has been successfully configured.

otheroptions
You can use other quorum-device-type-specific options. Refer to scconf_quorum_dev_scsi(1M) and scconf_quorum_dev_netapp_nas(1M) for details.

You need `solaris.cluster.quorum.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-S` *slm_options*
When used with the change form of the `sconf` command, sets properties to configure system resource control. If you do not assign a value to these properties, they are set

automatically to the default value. You can set the `-S` option on the Solaris 9 operation system but this configuration is only applied on Solaris 10 or later versions of this operating system.

The syntax for the `-S` option is:

```
–S [node=node] \
[,globalzoneshares=integer] \
[,defaultpsetmin=integer]
```

The `-S` option supports the following suboptions:

`globalzoneshares=`*globalzoneshares*
: Sets the number of shares that are assigned to the global zone. The lower limit for `globalzoneshares` is 1 and the upper limit is 65,535. To understand this upper limit, see the `prctl`(1) man page for information about the `zone.cpu-shares` attribute. The default value for `globalzoneshares` is 1. If, on a running cluster, there are no longer any online resource groups with CPU control configured in the global zone, the number CPU shares assigned to the global zone is set to the value of `globalzoneshares`.

`defaultpsetmin=`*defaultpsetmin*
: Sets the minimum number of CPU available in the default processor set. The default value is 1. The minimum value of `defaultpsetmin` is 1. Sun Cluster assigns a number of CPU as close as possible to the number you set for `defaultpsetmin` within the limit of available CPU. If the number assigned is lower than the number you requested Sun Cluster periodically attempts to assign the number of CPU you requested. This action might destroy some `dedicated_weak` processor sets. For information about `dedicated_weak` processor sets, see the scrgadm(1M) man page.

`node=`*node*
: Identifies nodes on which properties are to be set. Set these properties on each node you want to benefit from CPU control by specifying the name of the node. For each usage of the `-S` option, you can specify one node.

You need `solaris.cluster.node.modify` RBAC authorization to use this command option with `-c`. See `rbac`(5).

`-T` *authentication_options*
: Establishes authentication policies for nodes that are attempting to add themselves to the cluster configuration. Specifically, when a machine requests that it be added to the cluster as a cluster node (see scinstall(1M)), a check is made to determine whether or not the node has permission to join. If the node has permission, the joining node is authenticated. By default, any machine is allowed to add itself to the cluster.

  The `-T` *authentication_options* for each of the three forms of the command that accept `-T` are as follows:

  Add:

–T node=*nodename*[,...][,authtype=*authtype*]

Change:

–T authtype=*authtype*

Remove:

–T {node=*nodename*[,...] | all}

The -T option supports the following suboptions:

all
You can clear the list of all node names by specifying scconf -r -T all. A cleared authentication list means that any node can attempt to install and configure itself in the cluster.

node=*nodename*
Adds or removes hostnames from the list of nodes that are able to install and configure themselves as nodes in the cluster. At least one node suboption is required for the add form of the command and is optional for remove. If the authentication list is empty, any host can request that it be added to the cluster configuration. However, if the list has at least one name in it, all such requests are authenticated using the authentication list.

Illegal *nodename*s are accepted, including the node name of dot (.). The dot character is special in that if a *nodename* of . is added to the authentication list, all other names are removed. This feature prevents a host from attempting to install and configure itself in the cluster.

authtype=*authtype*
Is used with either the add or change form of the command.

The only currently supported authentication types (authtype) are des and sys (or unix). The default authentication type is sys, which provides the least amount of secure authentication.

When des, or Diffie-Hellman, authentication is used, entries should be added to the publickey database for each cluster node to be added before actually running the scinstall command to add the node.

You need solaris.cluster.node.modify RBAC authorization to use this command option with -a, -c, or -r. See rbac(5).

-v
When used with the -p option, requests a more verbose, or detailed, listing of the cluster configuration. If used with other options, additional information might be printed when an error is encountered.

You need solaris.cluster.device.read, solaris.cluster.transport.read, solaris.cluster.resource.read, solaris.cluster.node.read,

`solaris.cluster.quorum.read`, and `solaris.cluster.system.read` RBAC authorizations to use this command option with `-p`. See `rbac`(5).

`-w` *heartbeat_options*
Changes the global heartbeat parameters of a cluster, which effectively changes the heartbeat parameters across all the adapters of the cluster.

Sun Cluster relies on heartbeats over the private interconnect to detect communication failures among cluster nodes. Reducing the heartbeat timeout enables Sun Cluster to detect failures more quickly, as the time that is required to detect failures decreases when you decrease the values of heartbeat timeout. Thus, Sun Cluster recovers more quickly from failures, consequently increasing the availability of your cluster.

The `-w` option supports the following suboptions:

`heartbeat_quantum=`*quantum_milliseconds*
Defines how often to send heartbeats. Sun Cluster uses a 1 second (1,000 milliseconds) heartbeat quantum by default. Specify a value between 100 milliseconds and 10,000 milliseconds.

`heartbeat_timeout=`*timeout_milliseconds*
The time interval after which, if no heartbeats are received from the peer nodes, the corresponding path is declared as down. Sun Cluster uses a 10 second (10,000 millisecond) heartbeat timeout by default. Specify a value between 2,500 milliseconds and 60,000 milliseconds.

Note – Even under ideal conditions, when you reduce the values of heartbeat parameters with `-w`, there is always a risk that spurious path timeouts and node panics might occur. Always test and thoroughly qualify the lower values of heartbeat parameters under relevant workload conditions before actually implementing them in your cluster.

Usage With the `-w` option, you can change only one heartbeat suboption at a time. When decreasing the values of heartbeat parameters, change `heartbeat_quantum` first, followed by `heartbeat_timeout`. When increasing the values of heartbeat parameters, change `heartbeat_timeout` first, followed by `heartbeat_quantum`.

Note – The value you specify for `heartbeat_timeout` must always be greater than or equal to five times the value you specify for `heartbeat_quantum` (`heartbeat_timeout` >= (5*`heartbeat_quantum`)).

You need `solaris.cluster.system.modify` RBAC authorization to use `-w`. See `rbac`(5).

Examples **EXAMPLE 1** Decreasing the Heartbeat

The following example shows how to decrease the heartbeat quantum to 100 milliseconds from the Sun Cluster default of 1,000 milliseconds. This example also shows how to decrease the heartbeat timeout to 2500 milliseconds from the Sun Cluster default of 10,000 milliseconds.

EXAMPLE 1 Decreasing the Heartbeat *(Continued)*

```
phys-schost-1# scconf -c -w heartbeat_quantum=100
phys-schost-1# scconf -c -w heartbeat_timeout=2500
```

Because `heartbeat_timeout` must always be greater than or equal to five times `heartbeat_quantum`, you need to set `heartbeat_quantum` first. Otherwise, the requirement is not met. In other words, if `heartbeat_quantum` is currently set to the default 1,000 milliseconds, and if you were to set `heartbeat_timeout` to 2500 milliseconds, `heartbeat_timeout` would be *less* than five times `heartbeat_quantum`. The `scconf` command would consequently fail.

Once `heartbeat_quantum` is set to the correct value however, the requirement is maintained, and you can then set `heartbeat_timeout` to the decreased value.

EXAMPLE 2 Increasing the Heartbeat

The following example shows how to increase the heartbeat timeout and heartbeat quantum to Sun Cluster default values from the values to which you set these parameters in the previous example.

```
phys-schost-1# scconf -c -w heartbeat_timeout=10000
phys-schost-1# scconf -c -w heartbeat_quantum=1000
```

You set `heartbeat_timeout` first to maintain the requirement that `heartbeat_timeout` always be greater than or equal to five times `heartbeat_quantum`. Once `heartbeat_timeout` is set to the value you want, you can then set `heartbeat_quantum` to the new, increased value.

EXAMPLE 3 Typical Postinstallation Setup Operations

The following commands provide an example of a typical set of postinstallation setup operations that might be performed on a new two-node cluster. These commands add a shared quorum device to the cluster, clear `installmode`, configure a second set of cluster transport connections, and secure the cluster against other machines that might attempt to add themselves to the cluster:

```
phys-red# scconf -a -q globaldev=d0
phys-red# scconf -c -q reset
phys-red# scconf -a \
    -A trtype=dlpi,name=hme1,node=phys-red \
    -A trtype=dlpi,name=hme1,node=phys-green \
    -m endpoint=phys-red:hme1,endpoint=phys-green:hme1
phys-red# scconf -a -T node=.
```

Exit Status The following exit values are returned:

`0` The command completed successfully.

nonzero An error has occurred.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Interface Stability	Evolving

See Also Intro(1CL), cldevice(1CL), scconf_dg_rawdisk(1M), scconf_dg_svm(1M), scconf_dg_vxvm(1M), scconf_quorum_dev_scsi(1M), scconf_quorum_dev_netapp_nas(1M), scconf_transp_adap_bge(1M), scconf_transp_adap_ce(1M), scconf_transp_adap_e1000g(1M), scconf_transp_adap_eri(1M), scconf_transp_adap_ge(1M), scconf_transp_adap_hme(1M), scconf_transp_adap_ibd(1M), scconf_transp_adap_qfe(1M), scconf_transp_adap_sci(1M), scconf_transp_jct_dolphinswitch(1M), scconf_transp_jct_etherswitch(1M), scconf_transp_jct_ibswitch(1M), scdidadm(1M), scprivipadm(1M), `hosts`(4), `nsswitch.conf`(4), `publickey`(4), `attributes`(5), sctransp_dlpi(7p)

Warnings Use the `-w` option only when *all* nodes in a cluster are up. Do not use `-w` when any node in a cluster is down. Nodes might hang or panic as a result.

Clusters that contain one or more single-CPU nodes, or that contain more than eight nodes, are more likely to experience timeouts and node panics when the clusters run with low heartbeat parameter values.

Note – Even under ideal conditions, when you reduce the values of heartbeat parameters with `-w`, there is always a risk that spurious path timeouts and node panics might occur. Always test and thoroughly qualify the lower values of heartbeat parameters under relevant workload conditions before actually implementing them in your cluster.

Notes You should either back up the root file system on every node after changing the configuration with `scconf`, or keep a log of all changes. If you need to recover configuration changes between normal system backups, use the log to return to the most recent configuration.

Option lists specified with the `scconf` command are always executed in the order that you specify them on the command line. But, whenever possible, certain transport options (`-A`, `-B`, and `-m`) are processed by `scconf` as a single transaction against the cluster configuration database. Try to group all related options of this type together on a single command line to reduce overhead to the cluster.

Name scconf_dg_rawdisk – add, change or update raw-disk device group configuration

Synopsis

```
scconf -a -D type=rawdisk, [generic_options] [,globaldev=gdev1,globaldev=gdev1,...]
 [,localonly=true]

scconf -a -D type=rawdisk, [generic_options] [,globaldev=gdev1,globaldev=gdev1,...]
 [,localonly=true | false]

scconf -c -D name=diskgroup,autogen=true

scconf -r -D device_service_name [,nodelist=node[:node]...]
 [,globaldev=gdev1,...]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The following information is specific to the `scconf` command. To use the equivalent object-oriented commands, see the cldevicegroup(1CL) man page.

The `scconf_dg_rawdisk` utility adds, changes or updates raw-disk device group configuration

A raw disk is a disk that is not being used as part of a volume manager volume or metadevice. Raw-disk device groups allow you to define a set of disks within a disk device group.

At system boot, by default, a raw-disk device group is created for every Disk ID pseudo driver (DID) device in the configuration. By convention, the raw-disk device group names are assigned at initialization and are derived from the DID names. For every node added to a raw-disk disk device group, the `scconf` utility verifies that every device in the group is physically ported to the node.

The `scconf -a` (add) command can be used to create a raw-disk device group with multiple disk devices configured in it. A raw-disk device group is created for every disk device in the cluster at boot time.

Before you can add a new raw-disk device group, devices to be used in the new group must be removed from the device group created at boot time. Then a new raw-disk device group can be created containing these devices. This is accomplished by creating a list of these devices in the `globaldev` option of `scconf` along with a potential primary node preference list in the `nodelist` option. If the device group already exists, only new nodes and global devices will be added and nodes or devices which are part of an existing device group will be ignored.

If the `preferenced` suboption is not given with the `-a` option to create a new device group, then it is, by default, `false`. However, if the `preferenced` suboption is specified for the existing device group with a value of `true` or `false`, an error is returned. This is done in order to maintain the existing nodelist preference state.

If a device group should be mastered by only a particular node then it should be configured with the `otheroption` set to `localonly=true`. Only one node can be specified in the node list to create a `localonly` device group.

The `scconf -c` (change) command is used to change the order of the potential primary node preference, to enable or disable failback, to set the desired number of secondaries, and to add more global devices to the device group.

If you want to change the order of node preference list, then all the nodes currently existing in the device group must be specified in the `nodelist`. In addition, if you are changing the order of node preference, you must also set the `preferenced` suboption to `true`.

If the `preferenced` suboption is not specified with the change, the already established `true` or `false` setting is used.

New nodes cannot be added using the change form of the command. Change option can also be used for changing a device group to `localonly` device group and vice-versa. To change a device group to a `localonly` device group, set `otheroption` to `localonly=true`. Specify `localonly=false` to set it back to not the `localonly` device group. `nodelist` must already be set to a list of one node, or an error results. It is legal to specify a `nodelist` with the `change` form of the command, when you set `localonly` to `true`. This is, however, redundant, since the list can only contain the single node that is already configured. It would be an error to specify any other than the node that is already configured.

The `scconf -r` (remove) command can be used to remove the nodes, global devices, and the device group name from the cluster device-group configuration. If nodes or global devices are specified with the device-group name, they are removed from the device group first. After the last device and node are removed from the device group, the device group is also removed from cluster configuration. If only the name of the device group is given (no nodes or devices at all), the entire device group is removed.

If a raw-disk device name is registered in a raw-disk device group, then it cannot be registered in a Solaris Volume Manager device group or a VERITAS Volume Manager device group.

Options See the scconf(1M) man page for the list of supported generic options.

The following action options are used to describe the actions performed by the command. Only one action option is allowed in the command.

The following action options are supported:

`-a` Add a new raw-disk device group to the cluster configuration. You can also use this option to change the device group configuration.

`-c` Change the ordering of the node preference list, change preference and failback policy, change the desired number of secondaries, and also add more devices to the device group with the `globaldev` option. It is also used to set a device group as local only.

`-r` Remove the raw-disk device group name from the cluster.

The `autogen` flag is an indicator of the `scconf` command. This command does not list devices with the `autogen` property unless the `-v` command line option is used. When a device is used with the change form of the `scconf` command, the device's `autogen` property is reset, or set to false, unless `autogen=true` is also specified.

Examples

EXAMPLE 1 Using `scconf` Commands

The following `scconf` commands create a raw-disk device group, change the order of the potential primary nodes, change preference and failback policy, change the desired number of secondaries, and remove the raw-disk device group from the cluster configuration.

```
host1# scconf -a -D type=rawdisk,name=rawdisk_groupname,
nodelist=host1:host2:host3,preferenced=false,failback=enabled,
numsecondaries=,globaldev=d1,globaldev=d2

host1# scconf -a -D type=rawdisk,name=rawdisk_groupname,
nodelist=host1,globaldev=d1,globaldev=d2,localonly=true,
globaldev=d1,globaldev=d2

host1# scconf -c -D name=rawdisk_groupname,
nodelist=host3:host2:host1,preferenced=true,failback=disabled,
numsecondaries=2,globaldev=d4,globaldev=d5

host1# scconf -c -D name=rawdisk_groupname,localonly=true

host1# scconf -r -D name=rawdisk_groupname

host1# scconf -r -D name=rawdisk_groupname,nodelist=host1,host2
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cldevicegroup(1CL), scconf(1M), `attributes`(5)

Name scconf_dg_svm – change Solaris Volume Manager device group configuration.

Synopsis `scconf -c -D [`*generic_options*`]`

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The following information is specific to the `scconf` command. To use the equivalent object-oriented commands, see the cldevicegroup(1CL) man page.

A Solaris Volume Manager device group is defined by a name, the nodes upon which this group can be accessed, a global list of devices in the disk set, and a set of properties used to control actions such as potential primary preference and failback behavior.

For Solaris Volume Manager device groups, only one disk set can be assigned to a device group, and the group name must always match the name of the disk set itself.

In Solaris Volume Manager, a multihosted or shared device is a grouping of two or more hosts and disk drives that are accessible by all hosts, and that have the same device names on all hosts. This identical device naming requirement is achieved by using the raw disk devices to form the disk set. The device ID pseudo driver (DID) allows multihosted devices to have consistent names across the cluster. Only hosts already configured as part of a disk set itself can be configured into the `nodelist` of a Solaris Volume Manager device group. At the time drives are added to a shared disk set, they must not belong to any other shared disk set.

The Solaris Volume Manager `metaset` command creates the disk set, which also initially creates and registers it as a Solaris Volume Manager device group. Next, you must use the `scconf` command to set the node preference list, the `preferenced`, `failback` and `numsecondaries` suboptions.

If you want to change the order of node preference list or the failback mode, you must specify all the nodes that currently exist in the device group in the `nodelist`. In addition, if you are changing the order of node preference, you must also set the `preferenced` suboption to `true`.

If you do not specify the `preferenced` suboption with the "change" form of the command, the already established `true` or `false` setting is used.

You cannot use the `scconf` command to remove the Solaris Volume Manager device group from the cluster configuration. Use the Solaris Volume Manager `metaset` command instead. You remove a device group by removing the Solaris Volume Manager disk set.

Options See scconf(1M) for the list of supported generic options. See `metaset`(1M) for the list of `metaset` related commands to create and remove disk sets and device groups.

Only one action option is allowed in the command. The following action options are supported.

`-c` Change the ordering of the node preference list, change preference and failback policy, and change the desired number of secondaries.

Examples

EXAMPLE 1 Creating and Registering a Disk Set

The following `metaset` commands create the disk set `disksetand` register the disk set as a Solaris Volume Manager device group.

Next, the `scconf` command is used to specify the order of the potential primary nodes for the device group, change the preferenced and failback options, and change the desired number of secondaries.

```
host1# metaset -s diskset1 -a -h host1 host2

host1# scconf -c -D name=diskset1,nodelist=host2:host1,
preferenced=true,failback=disabled,numsecondaries=1
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cldevicegroup(1CL), scconf(1M), `metaset`(1M)

Name scconf_dg_vxvm – add, change, or update VxVM device group configuration.

Synopsis

```
scconf  -a -D type=vxvm,devicegroup-options[,localonly=true|false]

scconf  -c -D devicegroup-options[,sync]

scconf  -r -D name=devicegroupname
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The following information is specific to the `scconf` command. To use the equivalent object-oriented commands, see the cldevicegroup(1CL) man page.

The `scconf_dg_vxvm` command is used to add, change, and remove the VERITAS Volume Manager (VxVM) device groups to the Sun Cluster device-groups configuration.

Options See the scconf(1M) man page for the list of supported generic device-group options.

The following action options describe the actions that the command performs. Only one action option is allowed in the command.

`-a` Add a VxVM device group to the cluster configuration.

The `-a` (add) option adds a new VxVM device group to the Sun Cluster device-groups configuration. With this option you define a name for the new device group, specify the nodes on which this group can be accessed, and specify a set of properties used to control actions.

For VxVM device groups, you can only assign one VxVM disk group to a device group, and the device-group name must always match the name of the VxVM disk group. You cannot create a VxVM device group unless you first import the corresponding VxVM disk group on one of the nodes in that device's node list.

Before you can add a node to a VxVM device group, every physical disk in the disk group must be physically ported to that node. After you register the disk group as a VxVM device group, you must first deport the disk group from the current node owner and turn off the auto-import flag for the disk group.

To create a VxVM device group for a disk group, you must run the `scconf` command from the same node where the disk group was created.

`-c` Change the ordering of the node preference list, change preference and failback policy, and change the desired number of secondaries.

The `scconf -c` (change) option changes the order of the potential primary node preference, to enable or disable failback, to add more global devices to the device group, and to change the desired number of secondaries.

The `sync` suboption is used to synchronize the clustering software with VxVM disk-group volume information. The `sync` suboption is only valid with the change form of the command. Use the `sync` suboption whenever you add or remove a VxVM volume from a VxVM device group or change any volume attribute, such as owner, group, or access permissions.

Also use the `sync` suboption to change a device-group configuration to a replicated or non-replicated configuration.

For device groups that contain disks that use Hitachi TrueCopy data replication, this `sync` suboption synchronizes the device-group configuration and the replication configuration. This synchronization makes Sun Cluster software aware of disks that are configured for data replication and enables the software to handle failover or switchover as necessary.

After you create a Solaris Volume Manager disk set that contain disks that are configured for replication, you must run the `sync` suboption for the corresponding `svm` or `sds` device group. A Solaris Volume Manager disk set is automatically registered with Sun Cluster software as an `svm` or `sds` device group, but replication information is not synchronized at that time.

For newly created `vxvm` and `rawdisk` device-group types, you do not need to manually synchronize replication information for the disks. When you register a VxVM disk group or a raw-disk device group with Sun Cluster software, the software automatically discovers any replication information on the disks.

To change the order-of-node preference list from `false` to `true`, you must specify in the `nodelist` all the nodes that currently exist in the device group. You must also set the `preferenced` suboption to `true`.

If you do not specify the `preferenced` suboption with the `change` form of the command, the already established `true` or `false` setting is used.

If a disk group should be accessed by only one node, it should be configured with the `localonly` property set to `true`. This property setting puts the disk group outside the control of Sun Cluster software. Only one node can be specified in the node list to create a `localonly` disk group.

To change a local-only disk group to a regular VxVM disk group, set the `localonly` property to `false`.

`-r` Remove the specified VxVM device group from the cluster.

The `-r` (remove) option removes a VxVM device group from the Sun Cluster device-groups configuration. You can also use this form of command to remove the nodes from the VxVM device group configuration.

Examples

EXAMPLE 1 Using `scconf` Commands

The following `scconf` commands create a VxVM device group, change the order of the potential primary nodes, change the preference and failback policy for the device group, change the desired number of secondaries, and remove the VxVM device group from the cluster configuration.

```
host1# scconf -a -D type=vxvm,name=diskgrp1,nodelist=host1:host2:host3,\
preferenced=false,failback=enabled
host1# scconf -c -D name=diskgrp1,nodelist=host2:host1:host3,\
preferenced=true,failback=disabled,numsecondaries=2 sync
host1# scconf -r -D name=diskgrp1,nodelist=node1
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Evolving

See Also

Intro(1CL), cldevicegroup(1CL), scconf(1M), `attributes`(5)

Name scconf_quorum_dev_netapp_nas – add and remove shared Network Appliance network-attached storage (NAS) quorum devices and change various NAS cluster quorum configuration properties or states.

Synopsis

```
scconf {-a|-c|-r} -q name=devicename  otheroptions
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

A Network Appliance NAS device can be configured as a quorum device for Sun Cluster. The NAS configuration information consists of:

- a device name, which must be unique across quorum devices
- a filer name, which defaults to the device name if not specified
- a LUN ID, which defaults to `0` if not specified

To provide support for NAS devices as quorum devices, the administrator must install the quorum device support module provided by Network Appliance. If this module is not available, `scconf`prevents the addition of the quorum device. See *Sun Cluster With Network-Attached Storage Devices Manual for Solaris OS* for instructions about obtaining the support module.

Additionally, the iSCSI license must be valid for the Network Appliance device.

Options The following options can be used for NAS quorum devices. See scconf(1M) for the list of supported generic options. See scconf_quorum_dev_netapp_nas(1M) for options that are specific to shared disk quorum devices.

The `add` and `remove` forms of the command are used to add and remove NAS quorum devices to or from the configuration. The `change` form of the command is used for changing various properties of cluster quorum configuration.

Before you add a quorum device, you must set up and configure the device and the logical unit number (LUN) on the device to be configured as a quorum device. For detailed procedures, see your Network Appliance documentation. For Sun Cluster requirements of device setup, see *Sun Cluster With Network-Attached Storage Devices Manual for Solaris OS*. After the quorum device is added, you cannot change the `type`.

Add a NAS quorum device:

```
–q -a name=devicename,type=netapp_nas[,filer=filer-name][,lun_id=0]
```

Change a NAS quorum device's configuration:

```
–q -c name=devicename,{maintstate | reset}
```

Remove a NAS quorum device:

`–q -r name=`*devicename*

The `-q` option supports the following Network Appliance NAS-specific suboptions:

filer=*filer-name*
: Specifies the name of the device on the network that you can use to access the NAS device when you are using `rsh` or `telnet`.

lun_id=*0*
: Specifies the LUN ID on the NAS device that will be a NAS quorum device. The LUN ID defaults to `0`.

When `scconf` is interrupted or fails while performing quorum-related operations, quorum configuration information can become inconsistent in the cluster configuration database. If an inconsistency occurs, either run the same `scconf` command again or run it with the `reset` option to reset the quorum information.

Examples

EXAMPLE 1 Adding Network Appliance NAS Quorum Devices

The following `scconf` command adds the Network Appliance NAS quorum device `qd1`.

```
-a –q name=qd1,type=netapp_nas,filer=nas1.sun.com,lun_id=0
```

EXAMPLE 2 Removing Network Appliance NAS Quorum Devices

The following `scconf` command removes the Network Appliance NAS quorum device `qd1`.

```
-r –q name=qd1
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), clquorum(1CL), cluster(1CL), scconf(1M), scconf_quorum_dev_scsi(1M)

Name scconf_quorum_dev_quorum_server – add, remove, and configure a quorum server type of quorum device.

Synopsis `scconf [-q` *quorum-options*`]`

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

Sun Cluster provides the option of configuring a quorum server as a quorum device. This configuration information consists of a device name that must be unique across quorum devices, the address of the host machine on which the quorum server is running, and the port number on which the quorum server is listening for requests. If your cluster requires multiple quorum devices, configure multiple quorum servers or use storage devices for the additional quorum devices. A quorum server can act as only one quorum device for a cluster.

To configure the cluster to use a quorum server, the quorum server software must be installed, configured, and running on a machine that is accessible to all cluster nodes. The quorum server itself must be configured and running when this command is run on a cluster node. See `clquorumserver`(1M) for information about configuring the quorum server.

Options The following parameters are required for configuring a quorum server type of quorum device. See scconf(1M) for the list of supported generic options.

Use the `add` and `remove` forms of the command to add shared quorum devices to and remove shared quorum devices from the configuration file. Use the `change` form of the command to change various cluster quorum configuration properties or states. The following quorum server specific options can be used to change the cluster quorum configuration:

Add a Quorum Server Type of Quorum Device

Before adding a quorum device:

- The quorum server must be running on the quorum server host machine.
- You must enter the quorum server host name in the `/etc/inet/hosts` file.
- You must set the netmask for the quorum server host.

For information about the hosts file and netmask requirements, see the procedure on adding quorum server quorum devices in the *Sun Cluster System Administration Guide*. Once the quorum device is added, none of the parameters can be changed.

`# scconf –q -a name=`*devicename*`,type=quorum_server,qshost=`*qhost*`,port=`*portnumber*

`name=`*devicename*
: The name of a quorum server. This name must be unique among all quorum devices in the system.

`type=quorum_server`
: Indicates the type of disk device group to create. For a quorum server type of quorum device, the value of this parameter must be `quorum_server`.

`qhost=`*qhost*
The hostname of the machine on the network that can be reached by all cluster nodes and that is running the quorum server. Depending on the IPv4 or IPv6 configuration of the host, this hostname must have an entry in the `/etc/hosts` file, the `/etc/inet/ipnodes` file, or both.

`port=`*portnumber*
The port on which the quorum server listens for requests.

Note – If you need to change the port number of a quorum server while maintaining the same host name, remove the quorum device first, make your changes, then add the quorum device back.

Change the Configuration Parameters of a Quorum Server Type Quorum Device

```
# scconf -c -q name=devicename,maintstate | reset
```

If other parameters such as `qshost` or `port` must be changed, add a new quorum device with the new parameters and remove the existing quorum device.

Remove a Quorum Server Type of Quorum Device

```
# scconf -q name=devicename
```

When the `scconf` command is interrupted or fails while performing quorum-related operations, quorum configuration information can become inconsistent in the cluster configuration database. If this occurs, either run the same `scconf` command again or run the `scconf` command with the `reset` option to reset the quorum information.

Examples

EXAMPLE 1 Adding a Quorum Server Type of Quorum Device

The following `scconf` command adds a quorum server quorum device with its port number configured as 9000.

```
# scconf –q -a name=qd1,type=quorum_server,qshost=scclient1,port=9000
```

EXAMPLE 2 Removing a Quorum Server Type of Quorum Device

The following `scconf` command removes the quorum server quorum device named `qd1`.

```
# scconf -r –q name=qd1
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Stability	Evolving

See Also

Intro(1CL), clquorum(1CL), `clquorumserver`(1CL), cluster(1CL), scconf(1M), `gateways`(4), `hosts`(4)

Name scconf_quorum_dev_scsi – Add and remove shared SCSI quorum devices and change various SCSI cluster quorum configuration properties or states.

Synopsis

```
scconf {-a|-c|-r} -q globaldev=devicename  otheroptions

scconf {-a|-c|-r} -q name=devicename  otheroptions
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

A SCSI quorum device is considered to be any Sun Cluster supported, attached storage that is connected to two or more nodes of the cluster. The device must be managed by DID, and the device name that is provided must be a DID device name.

The SCSI quorum device has no other properties that can be specified.

Options The following options are specific to shared disk quorum devices. See scconf(1M) for the list of supported generic options. See scconf_quorum_dev_netapp_nas(1M) for options that are specific to NAS quorum devices.

The `add` and `remove` forms of the command are used to add and remove shared quorum devices to or from the configuration. The `change` form of the command is used for changing various properties of cluster quorum configuration. The `-q` *quorum-options* available for each of the three forms of the command can be used to change the cluster quorum configuration are as follows:

Add a shared quorum device:

```
–q -a globaldev=devicename[, node=node,node=node[, ...]]
```

or

```
–q -a name= devicename,type=scsi
```

or

```
 -q -a autoconfig[,noop]
```

Change a property or state of quorum configuration:

```
–q -c globaldev=devicename,{maintstate | reset}
```

or

```
-q -c autoconfig[,noop]
```

Remove a shared quorum device:

```
–q -r globaldev=devicename
```

or

`–q -r name=`*devicename*

`autoconfig`
When used with the `add` form of the command, automatically chooses and assigns one quorum device in the two-node cluster. The quorum device is chosen from the available devices. If a quorum device is already configured, the command aborts.

When used with the `change` form of the command, automatically chooses and assigns one device that replaces all existing quorum devices in the two-node cluster. The quorum device is chosen from the available devices.

All available devices in the cluster must be qualified to be a quorum device. The `autoconfig` suboption does not assess whether an available device is qualified to be a quorum device.

If the cluster contains more than two nodes, the `autoconfig` suboption makes no changes to the quorum configuration. Do not use the `autoconfig` suboption if you intend to configure a NAS device as quorum.

[,`noop`]
Is valid with the `autoconfig` suboption. The command prints to standard output the list of quorum devices that the `autoconfig` suboption would add or change. The `autoconfig,noop` suboption makes no changes to the quorum configuration.

When `scconf` is interrupted or fails while performing quorum-related operations, quorum configuration information can become inconsistent in the cluster configuration database. If an inconsistency occurs, either run the same `scconf` command again or run it with the `reset` option to reset the quorum information.

With the `add` form of the command, if a `name` is specified without a `node` list, the quorum device is added with a port defined for every node to which the device is attached. But, if a `node` list is given, at least two nodes must be provided, and each node in the list must be ported to the device.

Examples

EXAMPLE 1 Adding SCSI Quorum Devices

The following `scconf` commands adds a SCSI quorum device.

```
-a –q globaldev=/dev/did/rdsk/d4s2
    or
-a –q name=/dev/did/rdsk/d4s2,type=scsi
```

EXAMPLE 2 Changing SCSI Quorum Devices

The following `scconf` command changes a SCSI quorum device configuration.

EXAMPLE 2 Changing SCSI Quorum Devices *(Continued)*

```
-c -q globaldev=/dev/did/rdsk/d4s2,reset
    or
-c -q name=/dev/did/rdsk/d4s2,reset
```

EXAMPLE 3 Removing SCSI Quorum Devices

The following `scconf` command removes the SCSI quorum device. `qd1`.

```
-r –q globaldev=qd1
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), clquorum(1CL), cluster(1CL), scconf(1M), scconf_quorum_dev_netapp_nas(1M)

Name scconf_transp_adap_bge – configure the bge transport adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `bge` adapters as cluster transport adapters. These adapters can only be used with the `dlpi` transport type. The `bge` adapter is VLAN capable.

The `bge` adapter connects to a transport switch or to another `bge` adapter on a different node. In either case, the connection is made through a transport cable.

When a transport switch is used and the endpoints of the transport cable are configured by using the `scconf` command, the `scinstall` command, or other tools, you are asked to specify a port name on the transport switch. You can provide any port name, or accept the default, as long as the name is unique for the switch.

The default is to set the port name to the node ID hosting the adapter at the other end of the cable.

There are no user configurable properties for cluster transport adapters of this type.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scinstall(1M), `bge`(7D)

Name scconf_transp_adap_ce – configure the ce Sun Ethernet transport adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `ce` adapters as cluster transport adapters. These adapters can be used with transport types `dlpi`. The `ce` adapter is VLAN capable.

A `ce` adapter connects to a transport switch or to another `ce` adapter on a different node. In either case, the connection is made through a transport cable.

When a transport switch is used and the endpoints of the transport cable are configured by using `scconf`, `scinstall`, or other tools, you are asked to specify a port name on the transport switch. You can provide any port name, or accept the default, as long as the name is unique for the switch.

The default is to set the port name to the node ID hosting the adapter at the other end of the cable.

There are no user configurable properties for cluster transport adapters of this type.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scinstall(1M), ce(7D)

Name scconf_transp_adap_e1000g – configure the Intel PRO/1000 network adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `e1000g` Intel PRO/1000 network adapters as cluster transport adapters. These adapters can only be used with transport type `dlpi`.

The `e1000g` based network adapter connects to a transport switch or to another Ethernet adapter on a different node. In either case, the connection is made through a transport cable.

When a transport switch is used and the endpoints of the transport cable are configured by using `scconf`, `scinstall`, or other tools, you are asked to specify a port name on the transport switch. You can provide any port name, or accept the default, as long as the name is unique for the switch.

The default is to set the port name to the node identifier that hosts the adapter at the other end of the cable.

There are no user configurable properties for cluster transport adapters of this type.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scinstall(1M), `e1000g`(7D)

Name scconf_transp_adap_eri – configure the eri transport adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `eri` Ethernet adapters as cluster transport adapters. These adapters can only be used with transport type `dlpi`.

The `eri` Ethernet adapter connects to a transport switch or to another Ethernet adapter on a different node. In either case, the connection is made through a transport cable.

When a transport switch is used and the endpoints of the transport cable are configured by using `scconf`, `scinstall`, or other tools, you are asked to specify a port name on the transport switch. You can provide any port name, or accept the default, as long as the name is unique for the switch.

The default is to set the port name to the node ID hosting the adapter at the other end of the cable.

There are no user configurable properties for cluster transport adapters of this type.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scinstall(1M), `eri`(7D)

Name scconf_transp_adap_ge – configure the Gigabit Ethernet (ge) transport adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `ge` adapters as cluster transport adapters. These adapters can only be used with transport type `dlpi`.

The `ge` adapter connects to a transport switch or to another `ge` adapter on a different node. In either case, the connection is made through a transport cable.

When a transport switch is used and the endpoints of the transport cable are configured using `scconf`, `scinstall`, or other tools, you are asked to specify a port name on the transport switch. You can provide any port name, or accept the default, as long as the name is unique for the switch.

The default is to set the port name to the node `ID` hosting the adapter at the other end of the cable.

There are no user configurable properties for cluster transport adapters of this type.

See Also clinterconnect(1CL), clnode(1CL), scconf(1M), scinstall(1M), `ge`(7D)

Name scconf_transp_adap_hme – configure the hme transport adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `hme` Ethernet adapters as cluster transport adapters. These adapters can only be used with transport type `dlpi`.

The `hme` Ethernet adapter connects to a transport switch or to another Ethernet adapter on a different node. In either case, the connection is made through a transport cable.

When a transport switch is used and the endpoints of the transport cable are configured using `scconf`, `scinstall`, or other tools, you are asked to specify a port name on the transport switch. You can provide any port name, or accept the default, as long as the name is unique for the switch.

The default is to set the port name to the node ID hosting the adapter at the other end of the cable.

There are no user configurable properties for cluster transport adapters of this type.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scinstall(1M), `hme`(7D)

Name scconf_transp_adap_ibd – configure the InfiniBand (ibd) transport adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `ibd` adapters as cluster transport adapters. These adapters can only be used with transport type `dlpi`.

The `ibd` adapter connects to an InfiniBand transport switch. The connection is made through a transport cable.

When the endpoints of the transport cable are configured by using `scconf`, `scinstall`, or other tools, you are asked to specify a port name on the transport switch. You can provide any port name, or accept the default, as long as the name is unique for the switch.

The default is to set the port name to the node `ID` that hosts the adapter at the other end of the cable.

There are no user-configurable properties for cluster transport adapters of this type.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scconf_transp_jct_ibswitch(1M), scinstall(1M), `ibd`(7D)

Name scconf_transp_adap_qfe – configure the qfe transport adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `qfe` Ethernet adapters as cluster transport adapters. These adapters can only be used with transport type `dlpi`.

The `qfe` Ethernet adapter connects to a transport switch or to another Ethernet adapter on a different node. In either case, the connection is made through a transport cable.

When a transport switch is used and the endpoints of the transport cable are configured by using `scconf`, `scinstall`, or other tools, you are asked to specify a port name on the transport switch. You can provide any port name, or accept the default, as long as the name is unique for the switch.

The default is to set the port name to the node ID hosting the adapter at the other end of the cable.

There are no user configurable properties for cluster transport adapters of this type.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scinstall(1M), `qfe`(7d)

Name scconf_transp_adap_sci – configure the sci cluster transport adapter

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure `sci` SCI-PCI adapters as cluster transport adapters. These adapters can be used with the `dlpi` transport type.

The adapter name is `sci`*N*, for example, `sci0`. Do not use `scid`*N* as the adapter name.

An `sci` adapter can only be connected to another `sci` adapter or to an SCI switch. When an `sci` adapter is connected to an SCI switch, it is important that you specify the correct port name when referring to a port on the switch as an endpoint argument to the `scconf` or `scinstall` utility. The port name must match the port number on the SCI switch (the number that is printed on the switch itself). Failure to give the correct port name could result in the operation failing. The result of providing an incorrect port name will be the same as you would see if the cable between the adapter and the switch were removed.

There are no user-configurable properties for cluster transport adapters of this type.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scconf_transp_jct_dolphinswitch(1M), scinstall(1M)

Name scconf_transp_jct_dolphinswitch – configure the Dolphin cluster transport switch

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can use `SCI` switches as cluster transport switches, also called transport junctions. They are of switch type `switch`.

The Dolphin `SCI` switch is used with `SCI` adapters. The ports of a Dolphin `SCI` switch are numbered (printed on the switch itself). Use the port number as the name of the port. It is important that you specify the correct port name when referring to a port on the switch as an endpoint argument to `scconf` or `scinstall`. Failure to give the correct port name (which must be the same as the port number that appears on the switch), could result in the configuration operation failing or an operation running on a wrong port. This might bring down the cluster or prevent a node from coming up in clustered mode.

There are no user configurable properties on the Dolphin `SCI` switch.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL), scconf(1M), scconf_transp_adap_sci(1M), scinstall(1M)

Name scconf_transp_jct_etherswitch – configure an Ethernet cluster transport switch

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure Ethernet switches as cluster transport switches, also called transport junctions. They are of switch type `switch`. There are no user configurable properties.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL)

Name scconf_transp_jct_ibswitch – configure an InfiniBand cluster transport switch

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

You can configure InfiniBand switches as cluster transport switches, also called transport junctions. They are of switch type `switch`. There are no user configurable properties.

See Also Intro(1CL), clinterconnect(1CL), clnode(1CL),scconf_transp_adap_ibd(1M)

Name scdidadm – device identifier configuration and administration utility wrapper

Synopsis

```
/usr/cluster/bin/scdidadm -C

/usr/cluster/bin/scdidadm -c

/usr/cluster/bin/scdidadm -F {pathcount | scsi3 | useglobal} instance

/usr/cluster/bin/scdidadm -G

/usr/cluster/bin/scdidadm -G {pathcount | prefer3}

/usr/cluster/bin/scdidadm {-l | -L} [-h] [-o fmt]... [path | instance]

/usr/cluster/bin/scdidadm -R {path | instance | all}

/usr/cluster/bin/scdidadm -r

/usr/cluster/bin/scdidadm -T remote-nodename -e replication-type

/usr/cluster/bin/scdidadm -t source-instance:destination-instance -e replication-type
[-g replication-device-group]

/usr/cluster/bin/scdidadm [-u] [-i]

/usr/cluster/bin/scdidadm -v
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scdidadm` utility administers the device identifier (DID) pseudo device driver `did`.

The `scdidadm` utility performs the following primary operations:

- Creates driver configuration files
- Modifies entries in the file
- Loads the current configuration into the kernel
- Lists the mapping between device entries and `did` driver instance numbers

The startup script `/etc/init.d/bootcluster` uses the `scdidadm` utility to initialize the `did` driver. You can also use `scdidadm` to update or query the current device mapping between the devices present and the corresponding device identifiers and `did` driver instance numbers.

The `devfsadm`(1M) command creates the file system device entry points.

You can use this command only in the global zone.

Options The following options are supported:

`-C`
Removes all DID references to underlying devices that have been detached from the current node.

You can only use this option from a node that is booted in cluster mode. You can use this option only in the global zone.

Specify this option after the Solaris device commands have been used to remove references to nonexistent devices on the cluster nodes.

The `-F` option does not affect the fencing protocol of a configured quorum device.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-c`

Performs a consistency check against the kernel representation of the devices and the physical devices.

You can use this option only in the global zone.

On failing a consistency check, an error message is displayed. The process continues until all devices have been checked.

You need `solaris.cluster.device.read` RBAC authorization to use this command option. See `rbac`(5).

`-e` *type*

Specifies the replication type. When specifying the Hitachi replication type, this option can be used in combination with either the `-t` or `-T`option. When specifying the SRDF replication type, this option can only be used with the `-t` option.

`-F`

Overrides the global default fencing algorithm for individual specified devices.

The default fencing algorithm for a device can be set to one of the following values:

`pathcount`	Determines the fencing protocol by the number of DID paths that are attached to the shared device. ■ For a device that uses fewer than three DID paths, the command sets the SCSI-2 protocol. ■ For a device that uses three or more DID paths, the command sets the SCSI-3 protocol.
`scsi3`	Sets the SCSI-3 protocol. If the device does not support the SCSI-3 protocol, the fencing protocol setting remains unchanged.
`useglobal`	Sets the global default fencing setting for the specified devices.

By default, the global default fencing algorithm is set to `pathcount`. See the description of the `-G` option for information about setting the global default for fencing.

You can specify the device to modify by its instance number. The command accepts a space-delimited list of multiple devices. See the description of the `-o` option for more information about the `instance` form of device names.

The `-F` option does not affect the fencing protocol of a configured quorum device.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-G`

Sets or displays the current global default fencing algorithm for all shared devices.

When specified alone, the `-G` option displays the current global default fencing algorithm setting.

When specified with a setting value, the `-G` option sets the global default fencing to that value for all devices. The global default fencing can be set to one of the following values:

`prefer3` — Sets the SCSI-3 protocol for device fencing for all devices. The `pathcount` setting is assigned to any devices that do not support the SCSI-3 protocol.

`pathcount` — Determines the fencing protocol by the number of DID paths that are attached to the shared device.

- For devices that use fewer than three DID paths, the command sets the SCSI-2 protocol.
- For devices that use three or more DID paths, the command sets the SCSI-3 protocol.

By default, the global default fencing algorithm is set to `pathcount`.

The `-G` option does not affect the fencing protocol of a configured quorum device.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-g`

Specifies the replication device group.

`-h`

Prints a header when listing device mappings.

This option is meaningful only when used with the `-l` and `-L` options.

`-i`

Initializes the `did` driver.

You can use this option only in the global zone.

Use this option if you want to enable I/O requests to the `did` driver.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-L`
Lists all the paths, including those on remote hosts, of the devices in the DID configuration file.

You can use this option only in the global zone.

The output of this command can be customized using the `-o` option. When no `-o` options are specified, the default listing displays the *instance* number, all local and remote *fullpath* strings, and the *fullname*.

You need `solaris.cluster.device.read` RBAC authorization to use this command option. See `rbac`(5).

`-l`
Lists the local devices in the DID configuration.

You can use this option only in the global zone.

The output of this command can be customized using the `-o` option. When no `-o` options are specified, the default listing displays the *instance* number, the local *fullpath*, and the *fullname*.

You need `solaris.cluster.device.read` RBAC authorization to use this command option. See `rbac`(5).

`-o` *fmt*
Lists the devices currently known to the `did` driver according to the format specification *fmt*.

Multiple `-o` options can be specified. The *fmt* specification is interpreted as a comma-separated list of format option arguments. This option is meaningful only when used with the `-l` and `-L` options. The available format option arguments are the following:

`instance`	Prints the instance number of the device known by the `did` driver, for example, `1`.
`path`	Prints the physical path name of the device associated with this device identifier, for example, `/dev/rdsk/c0t3d0`.
`fullpath`	Prints the full physical path name of the device that is associated with this device identifier. This path name includes the host, for example, `phys-hostA:/dev/rdsk/c0t3d0`.

`host`	With the `-L` option, prints the names of all hosts that have connectivity to the specified device, one per line. With the `-l` option, prints the name of the local host that has connectivity to the specified device.
`name`	Prints the DID name of the device associated with this device identifier, for example, `d1`.
`fullname`	Prints the full DID path name of the device associated with this device identifier, for example, `/dev/did/rdsk/d1`.
`diskid`	Prints the hexadecimal representation of the device identifier associated with the instance of the device being listed.
`asciidiskid`	Prints the ASCII representation of the device identifier associated with the instance of the device being listed.
`defaultfencing`	Prints the default fencing algorithm set to the device.

`-R` {*path* | *instance* | all}
Performs a repair procedure on a particular device instance.

You can use this option only in the global zone.

The argument to this command can be either a particular physical device *path* that has been replaced with a new device, or the *instance* of the device that was just replaced. When used with the `all` keyword, the `scdidadm` utility updates the configuration data of all devices connected to the node.

You can only use this option from a node that is booted in cluster mode.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-r`
Reconfigures the database.

You can use this option only in the global zone.

When you specify this option, a thorough search of the `rdsk` and `rmt` device trees is conducted. A new instance number is assigned for all device identifiers that were not recognized before. A new path is added for each newly recognized device.

You can only use this option from a node that is booted in cluster mode.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-T` *remote-nodename*
Configures DID devices for use with the storage-based replication type you specify with the `-e` *replication—type* argument.

You can use this option only in the global zone.

Run this option from only one of the nodes configured with replicated devices. Use the *remote-nodename* option argument to specify the name of the remote node.

DID instances on the local node will be combined with the corresponding DID instance on the remote node, merging each pair of replicated devices into a single, logical DID device.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-t` *source-instance:destination-instance*
Moves the DID instance from the original source to a new destination instance that you specify with the *destination-instance* option argument.

You can use this option only in the global zone.

Use this option to move a DID instance back to its original location if the instance local was accidentally changed. After you run the command on all cluster nodes that are connected to the shared storage, run the `devfsadm` and `scgdevs` commands from one node to update the global-devices namespace with the configuration change.

If the *destination-instance* does not exist within the cluster, the DID device paths corresponding to the value of the *source-instance* argument are removed and recreated with the *destination-instance* you specify.

If the *destination-instance* already exists within the cluster, the path or paths for the *source-instance* are combined with the path or paths for the *destination-instance*, resulting in a single DID destination instance that contains all the paths for both instances.

Include the `-e` option to specify the replication type. If you are using an SRDF replication device, you must use the `-g` option to specify the replication device group.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-u`
Loads the device identifier configuration table into the kernel.

You can use this option only in the global zone.

This option loads all the currently known configuration information about device paths and their corresponding instance numbers into the kernel.

You need `solaris.cluster.device.modify` RBAC authorization to use this command option. See `rbac`(5).

`-v`
Prints the version number of this program.

You can use this option only in the global zone.

Examples

EXAMPLE 1 Adding Devices Attached to the Local Host to the CCR

```
# scdidadm -r
```

EXAMPLE 2 Listing the Physical Path of the Device

The following example lists the physical path of the device that corresponds to instance 2 of the `did` driver:

```
% scdidadm -l -o path 2
/dev/dsk/c1t4d0
```

EXAMPLE 3 Specifying Multiple Format Options

You can specify multiple format option arguments in either of the following ways:

```
% scdidadm –l –o path –o name 2
```

```
% scdidadm -l -o path,name 2
```

In either example, the output might look like this:

```
/dev/dsk/c1t4d0 d1
```

EXAMPLE 4 Configuring DID Devices for Use With Hitachi TrueCopy Storage-Based Replication

The following example configures a local DID device and a remote DID device for use with Hitachi TrueCopy storage-based replication. The command is run from a local source node that is configured with replicated devices. DID instances on the source node are combined with the corresponding DID instance on the remote destination node, `phys-schost-1`.

```
# scdidadm -T phys-schost-1 -e truecopy
```

EXAMPLE 5 Configuring DID Devices for Use With EMC SRDF Storage-Based Replication

The following example configures a local DID device and a remote DID device for use with EMC SRDF storage-based replication. The command is run from a local source node that is configured with replicated devices. DID instances on the source node are combined with the corresponding DID instance on the remote destination node, `phys-schost-1`.

```
# scdidadm -t 15:10 -e srdf -g devgroup1
```

EXAMPLE 6 Moving a DID Instance

The following example moves the DID instance on the source instance, `15`, to a new DID instance, `10`, then updates the configuration change in the global-devices namespace.

EXAMPLE 6 Moving a DID Instance *(Continued)*

```
# scdidadm -t 15:10
# devfsadm
# scgdevs
```

EXAMPLE 7 Performing a Repair Procedure

The following example performs the repair procedure for a particular device path. The device `/dev/dsk/c1t4d0` has been replaced with a new device with which a new device identifier is associated. The database is updated to show that this new device identifier corresponds to the instance number that was previously associated with the old device identifier:

```
# scdidadm -R c1t4d0
```

EXAMPLE 8 Performing a Repair Procedure

An alternative method of performing a repair procedure is to use the instance number associated with the device path. For example, if the instance number for the device `c1t4d0` in the previous example is `2`, then the following syntax performs the same operation as the previous example:

```
# scdidadm -R 2
```

EXAMPLE 9 Globally Setting the SCSI Protocol

The following example sets all SCSI devices in the cluster to the SCSI-3 protocol, except configured quorum devices and devices that do not support the SCSI-3 protocol. Any devices that do not support the SCSI-3 protocol are instead set to `pathcount`.

```
# scdidadm -G prefer3
```

EXAMPLE 10 Displaying the SCSI Protocol of a Single Device

The following example displays the SCSI protocol setting for the device `/dev/rdsk/c0t3d0`.

```
# scdidadm -L -o defaultfencing /dev/rdsk/c0t3d0
```

EXAMPLE 11 Setting the SCSI Protocol of a Single Device

The following example sets the device `11`, specified by instance number, to the SCSI-3 protocol. This device is not a configured quorum device and supports the SCSI-3 protocol.

```
# scdidadm -F scsi3 11
```

Exit Status

The following exit values are returned:

`0`	The command completed successfully.
`1`	An error occurred.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also Intro(1CL), cldevice(1CL), `devfsadm`(1M), scgdevs(1M), did(7)

Sun Cluster System Administration Guide for Solaris OS

Notes Each multiported tape drive or CD-ROM drive appears in the namespace once per physical connection.

Name scdpm – manage disk path monitoring daemon

Synopsis

```
scdpm [-a] {node | all}

scdpm -f filename

scdpm -m {[node | all][:/dev/did/rdsk/]dN | [:/dev/rdsk/]cNtXdY | all}

scdpm -n {node | all}

scdpm -p [-F] {[node | all][:/dev/did/rdsk/]dN | [/dev/rdsk/]cNtXdY | all}

scdpm -u {[node | all][:/dev/did/rdsk/]dN | [/dev/rdsk/]cNtXdY | all}
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scdpm` command manages the disk path monitoring daemon in a cluster. You use this command to monitor and unmonitor disk paths. You can also use this command to display the status of disk paths or nodes. All of the accessible disk paths in the cluster or on a specific node are printed on the standard output. You must run this command on a cluster node that is online and in cluster mode.

You can specify either a global disk name or a UNIX path name when you monitor a new disk path. Additionally, you can force the daemon to reread the entire disk configuration.

You can use this command only in the global zone.

Options The following options are supported:

`-a`

Enables the automatic rebooting of a node when all monitored disk paths fail, provided that the following conditions are met:

- All monitored disk paths on the node fail.
- At least one of the disks is accessible from a different node in the cluster.

You can use this option only in the global zone.

Rebooting the node restarts all resource and device groups that are mastered on that node on another node.

If all monitored disk paths on a node remain inaccessible after the node automatically reboots, the node does not automatically reboot again. However, if any monitored disk paths become available after the node reboots but then all monitored disk paths again fail, the node automatically reboots again.

You need `solaris.cluster.device.admin` role-based access control (RBAC) authorization to use this option. See `rbac`(5).

`-F`
If you specify the `-F` option with the `-p` option, `scdpm` also prints the faulty disk paths in the cluster. The `-p` option prints the current status of a node or a specified disk path from all the nodes that are attached to the storage.

`-f` *filename*
Reads a list of disk paths to monitor or unmonitor in *filename*.

You can use this option only in the global zone.

The following example shows the contents of *filename*.

```
u schost-1:/dev/did/rdsk/d5
m schost-2:all
```

Each line in the file must specify whether to monitor or unmonitor the disk path, the node name, and the disk path name. You specify the `m` option for monitor and the `u` option for unmonitor. You must insert a space between the command and the node name. You must also insert a colon (:) between the node name and the disk path name.

You need `solaris.cluster.device.admin` RBAC authorization to use this option. See `rbac`(5).

`-m`
Monitors the new disk path that is specified by *node*:*diskpath*.

You can use this option only in the global zone.

You need `solaris.cluster.device.admin` RBAC authorization to use this option. See `rbac`(5).

`-n`
Disables the automatic rebooting of a node when all monitored disk paths fail.

You can use this option only in the global zone.

If all monitored disk paths on the node fail, the node is *not* rebooted.

You need `solaris.cluster.device.admin` RBAC authorization to use this option. See `rbac`(5).

`-p`
Prints the current status of a node or a specified disk path from all the nodes that are attached to the storage.

You can use this option only in the global zone.

If you also specify the `-F` option, `scdpm` prints the faulty disk paths in the cluster.

Valid status values for a disk path are `Ok`, `Fail`, `Unmonitored`, or `Unknown`.

The valid status value for a node is `Reboot_on_disk_failure`. See the description of the `-a` and the `-n` options for more information about the `Reboot_on_disk_failure` status.

You need `solaris.cluster.device.read` RBAC authorization to use this option. See `rbac`(5).

`-u`
Unmonitors a disk path. The daemon on each node stops monitoring the specified path.

You can use this option only in the global zone.

You need `solaris.cluster.device.admin` RBAC authorization to use this option. See `rbac`(5).

Examples

EXAMPLE 1 Monitoring All Disk Paths in the Cluster Infrastructure

The following command forces the daemon to monitor all disk paths in the cluster infrastructure.

```
# scdpm -m all
```

EXAMPLE 2 Monitoring a New Disk Path

The following command monitors a new disk path.All nodes monitor `/dev/did/dsk/d3` where this path is valid.

```
# scdpm -m /dev/did/dsk/d3
```

EXAMPLE 3 Monitoring New Disk Paths on a Single Node

The following command monitors new paths on a single node. The daemon on the `schost-2` node monitors paths to the `/dev/did/dsk/d4` and `/dev/did/dsk/d5` disks.

```
# scdpm -m schost-2:d4 -m schost-2:d5
```

EXAMPLE 4 Printing All Disk Paths and Their Status

The following command prints all disk paths in the cluster and their status.

```
# scdpm -p
     schost-1:reboot_on_disk_failure  enabled
     schost-2:reboot_on_disk_failure  disabled
     schost-1:/dev/did/dsk/d4         Ok
     schost-1:/dev/did/dsk/d3         Ok
     schost-2:/dev/did/dsk/d4         Fail
     schost-2:/dev/did/dsk/d3         Ok
     schost-2:/dev/did/dsk/d5         Unmonitored
     schost-2:/dev/did/dsk/d6         Ok
```

EXAMPLE 5 Printing All Failed Disk Paths

The following command prints all of the failed disk paths on the `schost-2` node.

```
# scdpm -p -F all
     schost-2:/dev/did/dsk/d4         Fail
```

EXAMPLE 6 Printing the Status of All Disk Paths From a Single Node

The following command prints the disk path and the status of all disks that are monitored on the `schost-2` node.

```
# scdpm -p schost-2:all
     schost-2:reboot_on_disk_failure   disabled
     schost-2:/dev/did/dsk/d4          Fail
     schost-2:/dev/did/dsk/d3          Ok
```

Exit Status

The following exit values are returned:

`0` The command completed successfully.

`1` The command failed completely.

`2` The command failed partially.

Note – The disk path is represented by a node name and a disk name. The node name must be the host name or `all`. The disk name must be the global disk name, a UNIX path name, or `all`. The disk name can be either the full global path name or the disk name: `/dev/did/dsk/d3` or `d3`. The disk name can also be the full UNIX path name: `/dev/rdsk/c0t0d0s0`.

Disk path status changes are logged with the `syslogd` `LOG_INFO` facility level. All failures are logged with the `LOG_ERR` facility level.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Stability	Evolving

See Also

Intro(1CL), cldevice(1CL), clnode(1CL), `attributes`(5)

Sun Cluster System Administration Guide for Solaris OS

Name sceventmib – manage Sun Cluster event MIB module

Synopsis

```
sceventmib -a -c community -h host ...

sceventmib -a -t auth-type -u username [-f password-file]

sceventmib -d -s security-level -u username

sceventmib {-e | -n}

sceventmib -l protocol

sceventmib -m -t auth-type -u username

sceventmib -p {all | hosts | users}

sceventmib -r -c community -h host...

sceventmib -r -u username
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `sceventmib` command enables, disables, and configures the Sun Cluster event Management Information Base (MIB) module. When you issue this command on a cluster node, it affects only the configuration of the MIB module on that node. Each cluster node MIB module runs independently of others in the cluster.

You can use this command to enable or disable the MIB module on a cluster node. You can also use this command to set configuration properties, such as the version of SNMP trap notifications or the host name for an IP address to which to send trap notifications. The Sun Cluster Event MIB sends trap notifications on port 11162. The SNMP tree is viewed on port 11161.

You can use this command only in the global zone.

Options

Basic Options The following options direct the basic form and function of the command:

`-a`
Adds an entry for the specified SNMP host and community or for the specified user to the node configuration file.

You can use this option only in the global zone.

`-d`
Sets the default security level and user that you want used when you specify the `SNMPv3` protocol.

You can use this option only in the global zone.

You must specify a default user when you specify `SNMPv3`. SNMPv3 allows you to configure more than one user for the MIB module. Only a single default user can exist at any time. The first user that you add is automatically defined as the default user, regardless of the setting of this option. In this context, a default user is *not* necessarily the same as a Solaris OS user.

`-e`
Enables the Sun Cluster event MIB module on the node. This setting remains in effect until you change it, even after you reboot the node.

You can use this option only in the global zone.

`-l` *protocol*
Sets the version of the SNMP protocol to use with the MIBs.

You can use this option only in the global zone.

You can specify either `SNMPv2` or `SNMPv3` for the *protocol*. You cannot specify the `SNMPv3` protocol unless you have first configured at least one SNMPv3 user.

`-m`
Modifies the authentication type for an SNMP user.

You can use this option only in the global zone.

`-n`
Disables the Sun Cluster event MIB module on the node. This setting remains in effect until you change it, even after you reboot the node.

You can use this option only in the global zone.

`-p` {`all` | `hosts` | `users`}
Displays one of the following types of MIB configuration information:

`all` All MIB module configuration information

`hosts` Only the configuration information for SNMP hosts that are configured for use with the MIB module

`users` Only the configuration information for SNMP users who are configured to use the MIB module

You can use this option only in the global zone.

`-r`
Removes the entry for the specified SNMP host and community or for the specified SNMP user from the node configuration file.

You can use this option only in the global zone.

Additional Options

You can combine additional options with the basic options to modify the default behavior of each form of the command. Refer to the SYNOPSIS section for additional details about which of these options are legal with which forms of `sceventmib`.

The following additional options are supported:

`-c` *community*
Specifies the name of the SNMP community that you want to add to or remove from the node configuration file.

`-f` *password-file*
Specifies the name of a password file that contains one or more SNMP user names and their corresponding passwords.

Use the following syntax on every line that you include in the *password-file* file:

user:*password*

For example, specify the following lines for users Joseph Bloggs and Andrew Smith:

```
jbloggs:fgrxty_0
asmith:artfli!9
```

`-h` *host* …
Specifies the name of an SNMP host. You can specify either an IP address or a host name for *host*.

You can include a host in more than one community. However, if a host with the same name in the same community already exists, an error is returned.

`-s` *security-level*
Specifies the security level of the specified SNMPv3 user. This setting determines the degree to which the user can access the SNMP MIB module.

You can assign more than one security level to a user.

You specify one of the following **case-sensitive** settings for *security-level*:

`authNoPriv`	Authentication security measure is required, but privacy security measure is not required.
`authPriv`	Both authentication and privacy security measures are required.
`noAuthNoPriv`	Authentication and privacy security measures are not required.

`-t` *auth-type*
Specifies the authentication encryption mechanism that you want to use. You can specify either `MD5` or `SHA` for *auth-type*.

`-u` *username*
Specifies the name of an SNMPv3 user.

If you add an entry for a user and the same user name and security level already exists, the information is overwritten.

If you remove a default SNMPv3 user, the command automatically selects another default user.

Examples

EXAMPLE 1 Enabling the Event MIB

The following command enables the event MIB.

```
# sceventmib -e
```

EXAMPLE 2 Adding an SNMP Host to a Community

The following commands add a host to SNMP community `public`.

- The first example specifies the host by its host name, `sc-host`.

  ```
  # sceventmib -a -h sc-host -c public
  ```

- The second example specifies the host by its IP address, `10.0.0.25`.

  ```
  # sceventmib -a -h 10.0.0.25 -c public
  ```

EXAMPLE 3 Adding an SNMP User Without Providing a Password File

The following command adds the user `jbloggs` and specifies the MD5 authentication encryption mechanism. Because a password file is not specified, the command prompts the user to provide a password.

```
# sceventmib -a -t MD5 -u jbloggs
Enter password for user jbloggs: ********
```

EXAMPLE 4 Adding an SNMP User and Providing a Password File

The following command adds the user `jbloggs` and specifies the MD5 authentication encryption mechanism and the password file `pfile`. Because a password file is specified, the command does not prompt the user to provide a password.

```
# cat pfile
jbloggs:fgrxty_0
# sceventmib -a -f pfile -t MD5 -u jbloggs
```

EXAMPLE 5 Displaying All SNMP Configuration Information

The following command displays all SNMP configuration information.

```
# sceventmib -p all
```

EXAMPLE 6 Displaying Only Configuration Information About SNMP Hosts

The following command displays only configuration information about SNMP hosts.

```
# sceventmib -p hosts
```

EXAMPLE 7 Setting the Version of SNMP Protocol

The following command sets the SNMP protocol version to SNMPv3.

```
# sceventmib -l SNMPv3
```

EXAMPLE 8 Setting the Default SNMP User

The following command sets the default SNMP user to the user `jbloggs`, with authentication and privacy security measures required.

```
# sceventmib -d -s authPriv -u jbloggs
```

EXAMPLE 9 Modifying the Authentication Type of a User

The following command changes the authentication type of user `jbloggs` to SHA.

```
# sceventmib -m -t SHA -u jbloggs
```

EXAMPLE 10 Removing an SNMP Host

The following command removes the SNMP host with IP address `10.0.0.25` in community `public`.

```
# sceventmib -r -c public -h 10.0.0.25
```

EXAMPLE 11 Removing an SNMP User

The following command removes SNMP user `jbloggs`.

```
# sceventmib -r -u jbloggs
```

Exit Status

This command returns the following exit status codes:

`0`	The command completed successfully.
nonzero	An error occurred.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Interface Stability	Evolving

Files `/usr/cluster/lib/mib/sun-cluster-event-mib.mib`
Sun Cluster SNMP Event MIB definition file

See Also Intro(1CL), clsnmphost(1CL), clsnmpmib(1CL), clsnmpuser(1CL), `attributes`(5)

Sun Cluster System Administration Guide for Solaris OS

Name scgdevs – global devices namespace administration script

Synopsis `/usr/cluster/bin/scgdevs`

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scgdevs` command manages the global devices namespace. The global devices namespace is mounted under the `/global` directory and consists of a set of logical links to physical devices. As the `/dev/global` directory is visible to each node of the cluster, each physical device is visible across the cluster. This fact means that any disk, tape, or CD-ROM that is added to the global-devices namespace can be accessed from any node in the cluster.

The `scgdevs` command enables you to attach new global devices (for example, tape drives, CD-ROM drives, and disk drives) to the global-devices namespace without requiring a system reboot. You must run the `devfsadm` command before you run the `scgdevs` command.

Alternatively, you can perform a reconfiguration reboot to rebuild the global namespace and attach new global devices. See the `boot`(1M) man page for more information about reconfiguration reboots.

You must run this command from a node that is a current cluster member. If you run this command from a node that is not a cluster member, the command exits with an error code and leaves the system state unchanged.

You can use this command only in the global zone.

You need `solaris.cluster.system.modify` RBAC authorization to use this command. See the `rbac`(5) man page.

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`, `pfcsh`, or `pfksh` profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run the `su` command to assume a role. You can also use the `pfexec` command to issue privileged Sun Cluster commands.

Exit Status The following exit values are returned:

`0`	The command completed successfully.
nonzero	An error occurred. Error messages are displayed on the standard output.

Files

`/devices`	Device nodes directory
`/global/.devices`	Global devices nodes directory

`/dev/md/shared`	Solaris Volume Manager metaset directory

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also `pfcsh`(1), `pfexec`(1), `pfksh`(1), `pfsh`(1), Intro(1CL), cldevice(1CL), `boot`(1M), `devfsadm`(1M), `su`(1M), did(7)

Sun Cluster System Administration Guide for Solaris OS

Notes The `scgdevs` command, called from the local node, will perform its work on remote nodes asynchronously. Therefore, command completion on the local node does not necessarily mean that the command has completed its work clusterwide.

This document does not constitute an API. The `/global/.devices` directory and the `/devices` directory might not exist or might have different contents or interpretations in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API. This interface should be considered an unstable interface.

Name scinstall – initialize Sun Cluster software and establish new cluster nodes

Synopsis

media-mnt-pt/Solaris_*arch*/Product/sun_cluster/Solaris_*ver*/Tools/scinstall

media-mnt-pt/Solaris_*arch*/Product/sun_cluster/Solaris_*ver*/Tools/scinstall
-i [-k] [-s *srvc*[,...]] [-F [-C *clustername*] [-T *authentication-options*]
[-G [*special* | *mount-point*] [-o]] [-A *adapter-options*]
[-B *switch-options*] [-m *cable-options*] [-w *netaddr-options*]]

media-mnt-pt/Solaris_*arch*/Product/sun_cluster/Solaris_*ver*/Tools/scinstall
-i [-k] [-s *srvc*[,...]] [-N *cluster-member* [-C *clustername*]
[-G {*special* | *mount-point*}] [-A *adapter-options*]
[-B *switch-options*] [-m *cable-options*]]

media-mnt-pt/Solaris_*arch*/Product/sun_cluster/Solaris_*ver*/Tools/scinstall
-a *install-dir* [-d *dvdimage-dir*]

media-mnt-pt/Solaris_*arch*/Product/sun_cluster/Solaris_*ver*/Tools/scinstall
-c *jumpstart-dir* -h *nodename* [-d *dvdimage-dir*] [-s *srvc*[,...]]
[-F [-C *clustername*] [-G {*special* | *mount-point*}]]
[-T *authentication-options* [-A *adapter-options*] [-B *switch-options*]
[-m *cable-options*] [-w *netaddr-options*]]

media-mnt-pt/Solaris_*arch*/Product/sun_cluster/Solaris_*ver*/Tools/scinstall
-c *jumpstart-dir* -h *nodename* [-d *dvdimage-dir*] [-s *srvc*[,...]]
[-N *cluster-member* [-C *clustername*] [-G {*special* | *mount-point*}]
[-A *adapter-options*] [-B *switch-options*] [-m *cable-options*]]

media-mnt-pt/Solaris_*arch*/Product/sun_cluster/Solaris_*ver*/Tools/scinstall
-u *upgrade-mode*

/usr/cluster/bin/scinstall -u *upgrade-options*

/usr/cluster/bin/scinstall -r [-N *cluster-member*]
[-G *mount-point*]

scinstall -p [-v]

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scinstall` command performs a number of Sun Cluster node creation and upgrade tasks, as follows.

- The "initialize" form (`-i`) of `scinstall` establishes a node as a new Sun Cluster configuration member. It either establishes the first node in a new cluster (`-F`) or adds a node to an already-existing cluster (`-N`). Always run this form of the `scinstall` command from the node that is creating the cluster or is being added to the cluster.

- The “set up install server” form (`-a`) of `scinstall` creates an *install-dir* on any Solaris machine from which the command is run and then copies Sun Cluster installation media to that directory. Typically, you would create the target directory on an NFS server which has also been set up as a Solaris install server (see the `setup_install_server`(1M) man page).
- The “add install client” form (`-c`) of `scinstall` establishes the specified *nodename* as a custom JumpStart client in the *jumpstart-dir* on the machine from which the command is run. Typically, the *jumpstart-dir* is located on an already-established Solaris install server configured to JumpStart the Solaris *nodename* install client (see the `add_install_client`(1M) man page).
- The “remove” form (`-r`) of `scinstall` removes cluster configuration information and uninstalls Sun Cluster software from a cluster node.
- The “upgrade” form (`-u`) of `scinstall`, which has multiple modes and options, upgrades a Sun Cluster node. Always run this form of the `scinstall` command from the node being upgraded.
- The “print release” form (`-p`) of `scinstall` prints release and package versioning information for the Sun Cluster software that is installed on the node from which the command is run.

Without options, the `scinstall` command attempts to run in interactive mode.

Run all forms of the `scinstall` command other than the “print release” form (`-p`) as superuser.

The `scinstall` command is located in the `Tools` directory on the Sun Cluster installation media. If the Sun Cluster installation media has been copied to a local disk, *media-mnt-pt* is the path to the copied Sun Cluster media image. The `SUNWsczu` software package also includes a copy of the `scinstall` command.

Except for the `-p` option, you can run this command only from the global zone.

Options

Basic Options

The following options direct the basic form and function of the command.

None of the following options can be combined on the same command line.

`-a`

Specifies the “set up install server” form of the `scinstall` command. This option is used to create an *install-dir* on any Solaris machine from which the command is run and then make a copy of the Sun Cluster media in that directory.

You can use this option only in the global zone.

If the *install-dir* already exists, the `scinstall` command returns an error message. Typically, the target directory is created on an NFS server which has also been set up as a Solaris install server (see the `setup_install_server`(1M) man page).

`-c`

Specifies the "add install client" form of the `scinstall` command. This option establishes the specified *nodename* as a custom JumpStart client in the *jumpstart-dir* on the machine from which you issued the command.

You can use this option only in the global zone.

Typically, the *jumpstart-dir* is located on an already-established Solaris install server that is configured to JumpStart the *nodename* install client (see the `add_install_client`(1M) man page).

This form of the command enables fully-automated cluster installation from a JumpStart server by helping to establish each cluster node, or *nodename*, as a custom JumpStart client on an already-established Solaris JumpStart server. The command makes all necessary updates to the `rules` file in the specified *jumpstart-dir*. In addition, special JumpStart `class` files and `finish` scripts that support cluster initialization are added to the *jumpstart-dir*, if they are not already installed. Configuration data that is used by the Sun Cluster-supplied `finish` script is established for each node that you set up by using this method.

Users can customize the Solaris `class` file that the `-c` option to the `scinstall` command installs by editing the file directly in the normal way. However, it is always important to ensure that the Solaris `class` file defines an acceptable Solaris installation for a Sun Cluster node. Otherwise, the installation might need to be restarted.

Both the `class` file and `finish` script that are installed by this form of the command are located in the following directory:

jumpstart-dir`/autoscinstall.d/3.1`

The `class` file is installed as `autoscinstall.class`, and the `finish` script is installed as `autoscinstall.finish`.

For each cluster *nodename* that you set up with the `-c` option as an automated Sun Cluster JumpStart install client, this form of the command sets up a configuration directory as the following:

jumpstart-dir`/autoscinstall.d/nodes/`*nodename*

Options for specifying Sun Cluster node installation and initialization are saved in files that are located in these directories. Never edit these files directly.

You can customize the JumpStart configuration in the following ways:

- You can add a user-written `finish` script as the following file name:

  jumpstart-dir`/autoscinstall.d/nodes/`*nodename*`/finish`

  The `scinstall` command runs the user-written `finish` scripts after it runs the `finish` script supplied with the product.

- If the directory

  jumpstart-dir`/autoscinstall.d/nodes/`*nodename*`/archive`

  exists, the `scinstall` command copies all files in that directory to the new installation. In addition, if an `etc/inet/hosts` file exists in that directory, `scinstall` uses the hosts information found in that file to supply name-to-address mappings when a name service (NIS/NIS+/DNS) is not used.
- If the directory

  jumpstart-dir`/autoscinstall.d/nodes/`*nodename*`/patches`

  exists, the `scinstall` command installs all files in that directory by using the `patchadd`(1M) command. This directory is intended for Solaris software patches and any other patches that must be installed before Sun Cluster software is installed.

You can create these files and directories individually or as links to other files or directories that exist under *jumpstart-dir*.

See the `add_install_client`(1M) man page and related JumpStart documentation for more information about how to set up custom JumpStart install clients.

Run this form of the command from the *install-dir* (see the `-a` form of `scinstall`) on the JumpStart server that you use to initialize the cluster nodes.

Before you use the `scinstall` command to set up a node as a custom Sun Cluster JumpStart client, you must first establish each node as a Solaris install client. The JumpStart directory that you specify with the `-c` option to the `add_install_client` command should be the same directory that you specify with the `-c` option to the `scinstall` command. However, the `scinstall` *jumpstart-dir* does not have a server component to it, since you must run the `scinstall` command from a Solaris JumpStart server.

To remove a node as a custom Sun Cluster JumpStart client, simply remove it from the `rules` file.

`-i`

Specifies the "initialize" form of the `scinstall` command. This form of the command establishes a node as a new cluster member. The new node is the node from which you issue the `scinstall` command.

You can use this option only in the global zone.

If the `-F` option is used with `-i`, `scinstall` establishes the node as the first node in a new cluster.

If the `-o` option is used with the `-F` option, `scinstall` establishes a single-node cluster.

If the `-N` option is used with `-i`, `scinstall` adds the node to an already-existing cluster.

If the `-s` option is used and the node is an already-established cluster member, only the specified *srvc* (data service) is installed.

`-p`
Prints release and package versioning information for the Sun Cluster software that is installed on the node from which the command is run. This is the only form of `scinstall` that you can run as a non-superuser.

You can use this option in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone.

`-r`
Removes cluster configuration information and uninstalls Sun Cluster framework and data-service software from a cluster node. You can then reinstall the node or remove the node from the cluster. You must run the command on the node that you uninstall, from a directory that is not used by the cluster software. The node must be in noncluster mode.

You can use this option only in the global zone.

`-u` *upgrade-mode*
Upgrades Sun Cluster software on the node from which you invoke the `scinstall` command. The upgrade form of `scinstall` has multiple modes of operation, as specified by *upgrade-mode*. See `Upgrade Options` below for information specific to the type of upgrade that you intend to perform.

You can use this option only in the global zone.

Additional Options

You can combine additional options with the basic options to modify the default behavior of each form of the command. Refer to the `SYNOPSIS` section for additional details about which of these options are legal with which forms of the `scinstall` command.

The following additional options are supported:

`-d` *dvdimage-dir*
Specifies an alternate directory location for finding the media images of the Sun Cluster product and unbundled Sun Cluster data services.

If the `-d` option is not specified, the default directory is the media image from which the current instance of the `scinstall` command is started.

`-h` *nodename*
Specifies the node name. The `-h` option is only legal with the "add install client" (`-c`) form of the command.

The *nodename* is the name of the cluster node (that is, JumpStart install client) to set up for custom JumpStart installation.

`-k`
Specifies that `scinstall` will not install Sun Cluster software packages. The `-k` option is only legal with the "initialize" (`-i`) form of the command.

In Sun Cluster 3.0 and 3.1 software, if this option is not specified, the default behavior is to install any Sun Cluster packages that are not already installed. As of the Sun Cluster 3.2 release, this option is unnecessary. It is provided only for backwards compatibility with user scripts that use this option.

`-s` *srvc*[,...]
Specifies a data service. The `-s` option is only legal with the "initialize" (`-i`), "upgrade" (`-u`), or "add install client" (`-c`) forms of the command to install or upgrade the specified *srvc* (data service package).

If a data service package cannot be located, a warning message is printed, but installation otherwise continues to completion.

`-v`
Prints release information in verbose mode. The `-v` option is only legal with the "print release" (`-p`) form of the command to specify verbose mode.

In the verbose mode of "print release," the version string for each installed Sun Cluster software package is also printed.

`-F` [*config-options*]
Establishes the first node in the cluster. The `-F` option is only legal with the "initialize" (`-i`), "upgrade" (`-u`), or "add install client" (`-c`) forms of the command.

The establishment of secondary nodes will be blocked until the first node is fully instantiated as a cluster member and is prepared to perform all necessary tasks that are associated with adding new cluster nodes. If the `-F` option is used with the `-o` option, a single-node cluster is created and no additional nodes can be added during the cluster-creation process.

`-N` *cluster-member* [*config-options*]
Specifies the cluster member. The `-N` option is only legal with the "initialize" (`-i`), "add install client" (`-c`), "remove" (`-r`), or "upgrade" (`-u`) forms of the command.

- When used with the `-i`, `-c`, or `-u` option, the `-N` option is used to add additional nodes to an existing cluster. The specified *cluster-member* is typically the name of the first cluster node that is established for the cluster. However, the *cluster-member* can be the name of any cluster node that already participates as a cluster member. The node that is being initialized is added to the cluster of which *cluster-member* is already an active member. The process of adding a new node to an existing cluster involves updating the configuration data on the specified *cluster-member*, as well as creating a copy of the configuration database onto the local file system of the new node.
- When used with the `-r` option, the `-N` option specifies the *cluster-member*, which can be any other node in the cluster that is an active cluster member. The `scinstall` command contacts the specified *cluster-member* to make updates to the cluster configuration. If the `-N` option is not specified, `scinstall` makes a best attempt to find an existing node to contact.

Configuration Options

The *config-options* which can be used with the `-F` option or `-N` *cluster-member* option are as follows.

```
media-mnt-pt/Solaris_arch/Product/sun_cluster/Solaris_ver/Tools/scinstall
{-i |  -c  jumpstart-dir -h nodename}
[-F
   [ -C  clustername]
   [-G {special | mount-point} ]
   [-T authentication-options]
   [-A adapter-options]
   [-B switch-options]
   [-m endpoint=[this-node]:name[@port],endpoint=[node:]name[@port] ]
   [-o]
   [ -w  netaddr-options]
]
```

```
media-mnt-pt/Solaris_arch/Product/sun_cluster/Solaris_ver/Tools/scinstall
{-i |  -c  jumpstart-dir -h nodename}
[-N cluster-member
   [-C clustername]
   [-G {special | mount-point} ]
   [-A adapter-options]
   [-B switch-options]
   [-m endpoint=cable-options]
]
```

`-m` *cable-options*

Specifies the cluster interconnect connections. This option is only legal when the `-F` or `-N` option is also specified.

The `-m` option helps to establish the cluster interconnect topology by configuring the cables connecting the various ports found on the cluster transport adapters and switches. Each new cable configured with this form of the command establishes a connection from a cluster transport adapter on the current node to either a port on a cluster transport switch or an adapter on another node already in the cluster.

If you specify no `-m` options, the `scinstall` command attempts to configure a default cable. However, if you configure more than one transport adapter or switch with a given instance of `scinstall`, it is not possible for `scinstall` to construct a default. The default is to configure a cable from the singly-configured transport adapter to the singly-configured (or default) transport switch.

The `-m` *cable-options* are as follows.

```
-m endpoint=[this-node]:name[@port],endpoint=[node:]name[@port]
```

The syntax for the `-m` option demonstrates that at least one of the two endpoints must be an adapter on the node that is being configured. For that endpoint, it is not required to specify *this-node* explicitly. The following is an example of adding a cable:

```
-m endpoint=:hme1,endpoint=switch1
```

In this example, port 0 of the `hme1` transport adapter on this node, the node that `scinstall` is configuring, is cabled to a port on transport switch `switch1`. The port number that is used on `switch1` defaults to the node ID number of this node.

You must always specify two `endpoint` options with each occurrence of the `-m` option. The *name* component of the option argument specifies the name of either a cluster transport adapter or a cluster transport switch at one of the endpoints of a cable.

- If you specify the *node* component, the *name* is the name of a transport adapter.
- If you do not specify the *node* component, the *name* is the name of a transport switch.

If you specify no *port* component, the `scinstall` command attempts to assume a default port name. The default *port* for an adapter is always `0`. The default port *name* for a switch endpoint is equal to the node ID of the node being added to the cluster.

Refer to the individual cluster transport adapter and cluster transport switch man pages for more information regarding *port* assignments and other requirements. The man pages for cluster transport adapters use the naming convention `scconf_transp_adap_`*adapter*(1M). The man pages for cluster transport switches use the naming convention `scconf_transp_jct_`*switch*(1M).

Before you can configure a cable, you must first configure the adapters and/or switches at each of the two endpoints of the cable (see `-A` and `-B`).

`-o`

Specifies the configuration of a single-node cluster. This option is only legal when the `-i` and `-F` options are also specified.

Other `-F` options are supported but are not required. If the cluster name is not specified, the name of the node is used as the cluster name. You can specify transport configuration options, which will be stored in the CCR. The `-G` option is only required if the global-devices file system is not the default, `/globaldevices`. Once a single-node cluster is established, it is not necessary to configure a quorum device or to disable `installmode`.

`-w` *netaddr-options*

Specifies the network address for the private interconnect, or cluster transport. This option is only legal when the `-F` option is also specified.

Use this option to specify a private-network address for use on the private interconnect. You can use this option when the default private-network address collides with an address that is already in use within the enterprise. You can also use this option to customize the size of the IP address range that is reserved for use by the private interconnect. For more information, see the `networks`(4) and `netmasks`(4) man pages.

If not specified, the default network address for the private interconnect is `172.16.0.0`. The default netmask is `255.255.248.0`. This IP address range supports up to 64 nodes and 10 private networks.

The `-w` *netaddr-options* are as follows:

```
–w netaddr=netaddr[,netmask=netmask]
–w netaddr=netaddr[,maxnodes=nodes,maxprivatenets=maxprivnets]
–w netaddr=netaddr[,netmask=netmask,maxnodes=nodes,maxprivatenets=maxprivnets]
```

`netaddr=`*netaddr*
Specifies the private network address. The last two octets of this address must always be zero.

[`netmask=`*netmask*]
Specifies the netmask. The specified value must provide an IP address range that is greater than or equal to the default.

To assign a smaller IP address range than the default, specify the `maxnodes` and `maxprivatenets` operands.

[`,maxnodes=`*nodes*`,maxprivatenets=`*maxprivnets*]
Specifies the maximum number of nodes and private networks that the cluster is ever expected to have. The command uses these values to calculate the minimum netmask that the private interconnect requires to support the specified number of nodes and private networks. The maximum value for *nodes* is 64 and the minimum value is 2. The maximum value for *maxprivnets* is 128 and the minimum value is 2.

[`,netmask=`*netmask*`,maxnodes=`*nodes*`,maxprivatenets=`*maxprivnets*]
Specifies the netmask and the maximum number of nodes and private networks that the cluster is ever expected to have. You must specify a netmask that can sufficiently accommodate the specified number of *nodes* and *privnets*. The maximum value for *nodes* is 64 and the minimum value is 2. The maximum value for *privnets* is 128 and the minimum value is 2.

If you specify only the `netaddr` suboption, the command assigns the default netmask of `255.255.248.0`. The resulting IP address range accommodates up to 64 nodes and 10 private networks.

To change the private-network address or netmask after the cluster is established, use the `cluster` command or the `clsetup` utility.

`-A` *adapter-options*
Specifies the transport adapter and, optionally, its transport type. This option is only legal when the `-F` or `-N` option is also specified.

Each occurrence of the `-A` option configures a cluster transport adapter that is attached to the node from which you run the `scinstall` command.

If no `-A` options are specified, an attempt is made to use a default adapter and transport type. The default transport type is `dlpi`. In Sun Cluster 3.2 for SPARC, the default adapter is `hme1`.

When the adapter transport type is `dlpi`, you do not need to specify the `trtype` suboption. In this case, you can use either of the following two forms to specify the `-A` *adapter-options*:

```
-A [trtype=type,]name=adaptername[,vlanid=vlanid][,other-options]
-A adaptername
```

[trtype=*type*]
: Specifies the transport type of the adapter. Use the `trtype` option with each occurrence of the `-A` option for which you want to specify the transport type of the adapter. An example of a transport `type` is `dlpi` (see the sctransp_dlpi(7p) man page).

  The default transport type is `dlpi`.

name=*adaptername*
: Specifies the adapter name. You must use the `name` suboption with each occurrence of the `-A` option to specify the *adaptername*. An *adaptername* is constructed from a *device name* that is immediately followed by a *physical-unit* number, for example, `hme0`.

  If you specify no other suboptions with the `-A` option, you can specify the *adaptername* as a standalone argument to the `-A` option, as `-A` *adaptername*.

vlanid=*vlanid*
: Specifies the VLAN ID of the tagged-VLAN adapter.

[*other-options*]
: Specifies additional adapter options. When a particular adapter provides any other options, you can specify them by using the `-A` option. Refer to the individual Sun Cluster man page for the cluster transport adapter for information about any special options that you might use with the adapter.

`-B` *switch-options*
: Specifies the transport switch, also called transport junction. This option is only legal when the `-F` or `-N` option is also specified.

  Each occurrence of the `-B` option configures a cluster transport switch. Examples of such devices can include, but are not limited to, Ethernet switches, other switches of various types, and rings.

  If you specify no `-B` options, `scinstall` attempts to add a default switch at the time that the first node is instantiated as a cluster node. When you add additional nodes to the cluster, no additional switches are added by default. However, you can add them explicitly. The default switch is named `switch1`, and it is of type `switch`.

  When the switch type is type `switch`, you do not need to specify the `type` suboption. In this case, you can use either of the following two forms to specify the `-B` *switch-options*.

```
-B [type=type,]name=name[,other-options]
-B name
```

If a cluster transport switch is already configured for the specified switch *name*, `scinstall` prints a message and ignores the `-B` option.

If you use directly-cabled transport adapters, you are not required to configure any transport switches. To avoid configuring default transport switches, use the following special `-B` option:

```
-B type=direct
```

[type=*type*]
Specifies the transport switch type. You can use the `type` option with each occurrence of the `-B` option. Ethernet switches are an example of a cluster transport switch which is of the switch type `switch`. See the individual Sun Cluster man page for the cluster transport switch for more information.

You can specify the `type` suboption as `direct` to suppress the configuration of any default switches. Switches do not exist in a transport configuration that consists of only directly connected transport adapters. When the `type` suboption is set to `direct`, you do not need to use the `name` suboption.

name=*name*
Specifies the transport switch name. Unless the `type` is `direct`, you must use the `name` suboption with each occurrence of the `-B` option to specify the transport switch *name*. The *name* can be up to 256 characters in length and is made up of either letters or digits, with the first character being a letter. Each transport switch name must be unique across the namespace of the cluster.

If no other suboptions are needed with `-B`, you can give the switch *name* as a standalone argument to `-B` (that is, `-B` *name*).

[*other-options*]
Specifies additional transport switch options. When a particular switch type provides other options, you can specify them with the `-B` option. Refer to the individual Sun Cluster man page for the cluster transport switch for information about any special options that you might use with the switches.

`-C` *clustername*
Specifies the name of the cluster. This option is only legal when the `-F` or `-N` option is also specified.

- If the node that you configure is the first node in a new cluster, the default *clustername* is the same as the name of the node that you are configuring.
- If the node that you configure is being added to an already-existing cluster, the default *clustername* is the name of the cluster to which *cluster-member* already belongs.

It is an error to specify a *clustername* that is not the name of the cluster to which *cluster-member* belongs.

`-G` {*special* | *mount-point*}
Specifies a raw *special* disk device or a file system for the global-devices mount point. This option is only legal when the `-F`, `-N`, or `-r` option is also specified.

- When used with the `-F` or `-N` option, the `-G` option specifies the raw *special* disk device or the file system *mount-point* to use in place of the `/globaldevices` mount point. Each cluster node must have a local file system that is mounted globally on `/global/.devices/node@`*nodeID* before the node can successfully participate as a cluster member. However, since the node ID is not known until the `scinstall` command is run, `scinstall` attempts to add the necessary entry to the `vfstab`(4) file when it does not find a `/global/.devices/node@`*nodeID* mount.

  By default, the `scinstall` command looks for an empty file system that is mounted on `/globaldevices`. If such a file system is provided, the `scinstall` command makes the necessary changes to the `vfstab` file. These changes create a new `/global/.devices/node@`*nodeID* mount point and remove the default `/globaldevices` mount point. However, if `/global/.devices/node@`*nodeID* is not mounted and an empty `/globaldevices` file system is not provided, the `-G` option must be used to specify the raw *special* disk device or the file system *mount-point* to use in place of `/globaldevices`.

  If a raw *special* disk device name is specified and `/global/.devices/node@`*nodeID* is not mounted, a file system is created on the device by using the `newfs` command. It is an error to supply the name of a device with an already-mounted file system.

  As a guideline, this file system should be at least 512 Mbytes in size. If this partition or file system is not available, or is not large enough, it might be necessary to reinstall the Solaris operating environment.
- When used with the `-r` option, the `-G` *mount-point* option specifies the new mount-point name to use to restore the former `/global/.devices` mount point. If the `-G` option is not specified, the mount point is renamed `/globaldevices` by default.

`-T` *authentication-options*
Specifies node-authentication options for the cluster. This option is only legal when the `-F` option is also specified.

Use this option to establish authentication policies for nodes that attempt to add themselves to the cluster configuration. Specifically, when a machine requests that it be added to the cluster as a cluster node, a check is made to determine whether or not the node has permission to join. If the joining node has permission, it is authenticated and allowed to join the cluster.

You can only use the `-T` option with the `scinstall` command when you set up the very first node in the cluster. If the authentication list or policy needs to be changed on an already-established cluster, use the `scconf` command.

The default is to allow any machine to add itself to the cluster.

The `-T` *authentication-options* are as follows.

`-T node=`*nodename*`[,...][,authtype=`*authtype*`]`

node=*nodename*[,…]
: Specifies node names to add to the node authentication list. You must specify at least one `node` suboption to the `-T` option. This option is used to add node names to the list of nodes that are able to configure themselves as nodes in the cluster. If the authentication list is empty, any node can request that it be added to the cluster configuration. However, if the list has at least one name in it, all such requests are authenticated by using the authentication list. You can modify or clear this list of nodes at any time by using the `scconf` command or the `clsetup` utility from one of the active cluster nodes.

[`authtype=`*authtype*]
: Specifies the type of node authentication. The only currently supported `authtypes` are `des` and `sys` (or `unix`). If no `authtype` is specified, `sys` is the default.

  If you will you specify `des` (Diffie-Hellman) authentication, first add entries to the `publickey`(4) database for each cluster node to be added, before you run the `-T` option to the `scinstall`command.

  You can change the authentication type at any time by using the `scconf` command or the `clsetup` utility from one of the active cluster nodes.

Upgrade Options

The `-u` *upgrade-modes* and the *upgrade-options* for standard (nonrolling) upgrade, rolling upgrade, live upgrade, and dual-partition upgrade are as follows.

Standard (Nonrolling), Rolling, and Live Upgrade

Use the `-u update` mode to upgrade a cluster node to a later Sun Cluster software release in standard (nonrolling), rolling, or live upgrade mode.

- A standard, or nonrolling, upgrade process takes a cluster node out of production and upgrades the node to a later Sun Cluster software release.
- Live upgrade mode works in conjunction with Solaris Live Upgrade to upgrade an inactive boot environment while your cluster node continues to serve cluster requests. Once the upgrade is complete, you will activate the new boot environment and reboot the node.
- A rolling upgrade process takes only one cluster node out of product at a time. This process can only be used to upgrade Solaris and Sun Cluster software to an update release of the versions that are already installed. While you upgrade one node, cluster services continue on the rest of the cluster nodes. After a node is upgraded, you bring it back into the cluster and repeat the process on the next node to upgrade. After all nodes are upgraded, you must run the `scversions` command on one cluster node to commit the cluster to the upgraded version. Until this command is run, some new functionality that is introduced in the update release might not be available.

The *upgrade-options* to `-u update` for standard and rolling mode are as follows.

```
media-mnt-pt/Solaris_arch/Product/sun_cluster/Solaris_ver/Tools/scinstall -u update
    [-s {srvc[,...] | all}] [-d dvdimage-dir] [ -O ]
    [-S {interact | testaddr=testipaddr@adapter[,testaddr=...]} ]
```

For live upgrade mode, also use the `-R` *BE-mount-point* option to specify the inactive boot environment. The *upgrade-options* to `-u update` for live upgrade mode are as follows.

```
media-mnt-pt/Solaris_arch/Product/sun_cluster/Solaris_ver/Tools/scinstall -u update
    -R BE-mount-point [-s {srvc[,...] | all}] [-d dvdimage-dir] [ -O ]
    [-S {interact | testaddr=testipaddr@adapter[,testaddr=...]} ]
```

`-R` *BE-mount-point*
: Specifies the root for an inactive boot environment. This is the mount point that is specified to the `lumount` command. This option is required if you are performing a live upgrade.

`-s` {*srvc*[,…] | all}
: Upgrades data services. If the `-s` option is not specified, only cluster framework software is upgraded. If the `-s` option is specified, only the specified data services are upgraded.

  The `-s` option is not compatible with the `-S` test IP address option.

  The following suboptions to the `-s` option are specific to the `update` mode of upgrade:

  `all`
  : Upgrades all data services.

    This suboption to `-s` is only legal with the `update` mode.

    This suboption upgrades all data services currently installed on the node, except those data services for which an update version does not exist in the update release.

  srvc
  : Specifies the upgrade name of an individual data service.

    The value of *srvc* for a data service can be derived from the `CLUSTER` entry of the `.clustertoc` file for that data service. The `.clustertoc` file is located in the *media-mnt-pt*`/components/`*srvc*`/Solaris_`*ver*`/Packages/` directory of the data service software. The `CLUSTER` entry takes the form `SUNWC_DS_`*srvc*. For example, the value of the `CLUSTER` entry for the Sun Cluster HA for NFS data service is `SUNWC_DS_nfs`. To upgrade only the Sun Cluster HA for NFS data service, you issue the command `scinstall -u update -s nfs`, where `nfs` is the upgrade name of the data service.

    The following is a list of Sun Cluster data services and the upgrade name to specify to the `-s` option. For a data service that can be upgraded with the `scinstall` command but that is not included in this list, see the `.clustertoc` file of the data service software to derive the upgrade name to use.

Data Service	Upgrade Name
Sun Cluster HA for Agfa IMPAX	pax
Sun Cluster HA for Apache	apache
Sun Cluster HA for Apache Tomcat	tomcat
Sun Cluster HA for BEA WebLogic Server	wls
Sun Cluster HA for DHCP	dhcp
Sun Cluster HA for DNS	dns
Sun Cluster HA for Kerberos	krb5
Sun Cluster HA for MySQL	mys
Sun Cluster HA for N1 Grid Engine	n1ge
Sun Cluster HA for N1 Grid Service Provisioning System	n1sps
Sun Cluster HA for NFS	nfs
Sun Cluster HA for Oracle	oracle
Sun Cluster HA for Oracle E-Business Suite	ebs
Sun Cluster HA for PostgreSQL	PostgreSQL
Sun Cluster HA for Samba	smb
Sun Cluster HA for SAP	sap
Sun Cluster HA for SAP DB	sapdb
Sun Cluster HA for SAP liveCache	livecache
Sun Cluster HA for SAP Web Application Server	sapwebas
Sun Cluster HA for Siebel	siebel
Sun Cluster HA for Solaris Containers	container
Sun Cluster HA for Sun Java System Application Server	s1as
Sun Cluster HA for Sun Java System Application Server EE (HADB)	hadb
Sun Cluster HA for Sun Java System Message Queue	s1mq
Sun Cluster HA for Sun Java System Web Server	iws
Sun Cluster HA for SWIFTAlliance Access	saa
Sun Cluster HA for SWIFTAlliance Gateway	sag
Sun Cluster HA for Sybase ASE	sybase

Data Service	Upgrade Name
Sun Cluster HA for WebSphere MQ	mqs
Sun Cluster HA for WebSphere MQ Integrator	mqi
Sun Cluster Oracle Application Server (9i)	9ias
Sun Cluster Support for Oracle Real Application Clusters	oracle_rac

`-O`
Overrides the hardware validation and bypasses the version-compatibility checks.

`-S` {interact | testaddr=*testipaddr@adapter*[,testaddr=…]
Specifies test IP addresses. This option allows the user either to direct the command to prompt the user for the required IP network multipathing (IPMP) addresses or to supply a set of IPMP test addresses on the command line for the conversion of NAFO to IPMP groups. See Chapter 30, "Introducing IPMP (Overview)," in *System Administration Guide: IP Services* for additional information about IPMP.

It is illegal to combine both the `interact` and the `testaddr` suboptions on the same command line.

Note – The `-S` option is only required when one or more of the NAFO adapters in `pnmconfig` is not already converted to use IPMP.

The suboptions of the `-S` option are the following:

`interact`
Prompt the user to supply one or more IPMP test addresses individually.

`testaddr=`*testipaddr*`@`*adapter*
Directly specify one or more IPMP test addresses.

testipaddr The IP address or hostname, in the `/etc/inet/hosts` file, that will be assigned as the routable, no-failover, deprecated test IP address to the adapter. IPMP uses test addresses to detect failures and repairs. See "IPMP Addressing" in *System Administration Guide: IP Services* for additional information about configuring test IP addresses.

adapter The name of the NAFO network adapter to add to an IPMP group.

Dual-Partition Upgrade

Use the `-u` *upgrade-modes* and *upgrade-options* for dual-partition upgrade to perform the multiple stages of a dual-partition upgrade. The dual-partition upgrade process first involves assigning cluster nodes into two groups, or partitions. Next, you upgrade one partition while the other partition provides cluster services. You then switch services to the upgraded partition, upgrade the remaining partition, and rejoin the upgraded nodes of the second

partition to the cluster formed by the upgraded first partition. The *upgrade-modes* for dual-partition upgrade also include a mode for recovery after a failure during a dual-partition upgrade.

Dual-partition upgrade modes are used in conjunction with the `-u update` upgrade mode. See the upgrade chapter of the *Sun Cluster Software Installation Guide for Solaris OS* for more information.

The *upgrade-modes* and *upgrade-options* to `-u` for dual-partition upgrade are as follows:

media-mnt-pt`/Solaris_`*arch*`/Product/sun_cluster/Solaris_`*ver*`/Tools/scinstall \`
` -u begin -h` *nodelist*

media-mnt-pt`/Solaris_`*arch*`/Product/sun_cluster/Solaris_`*ver*`/Tools/scinstall -u plan`

media-mnt-pt`/Solaris_`*arch*`/Product/sun_cluster/Solaris_`*ver*`/Tools/scinstall -u recover`

media-mnt-pt`/Solaris_`*arch*`/Product/sun_cluster/Solaris_`*ver*`/Tools/scinstall -u status`

`/usr/cluster/bin/scinstall -u apply`

`/usr/cluster/bin/scinstall -u status`

`apply`
Specifies that upgrade of a partition is completed. Run this form of the command from any node in the upgraded partition, after all nodes in that partition are upgraded.

The `apply` upgrade mode performs the following tasks:

First partition
When run from a node in the first partition, the `apply` upgrade mode prepares all nodes in the first partition to run the new software.

When the nodes in the first partition are ready to support cluster services, the command remotely executes the scripts `/etc/cluster/ql/cluster_pre_halt_apps` and `/etc/cluster/ql/cluster_post_halt_apps` that are on the nodes in the second partition. These scripts are used to call user-written scripts that stop applications that are not under Resource Group Manager (RGM) control, such as Oracle Real Application Clusters (RAC).

- The `cluster_pre_halt_apps` script is run before applications that are under RGM control are stopped.
- The `cluster_post_halt_apps` script is run after applications that are under RGM control are stopped, but before the node is halted.

Note – Before you run the `apply` upgrade mode, modify the script templates as needed to call other scripts that you write to stop certain applications on the node. Place the modified scripts and the user-written scripts that they call on each node in the first partition. These scripts are run from one arbitrary node in the first partition. To stop applications that are running on more than one node in the first partition, modify the user-written scripts accordingly. The unmodified scripts perform no default actions.

After all applications on the second partition are stopped, the command halts the nodes in the second partition. The shutdown initiates the switchover of applications and data services to the nodes in the first partition. Then the command boots the nodes in the second partition into cluster mode.

If a resource group was offline because its node list contains only members of the first partition, the resource group comes back online. If the node list of a resource group has no nodes that belong to the first partition, the resource group remains offline.

Second partition

When run from a node in the second partition, the `apply` upgrade mode prepares all nodes in the second partition to run the new software. The command then boots the nodes into cluster mode. The nodes in the second partition rejoin the active cluster that was formed by the nodes in the first partition.

If a resource group was offline because its node list contains only members of the second partition, the resource group comes back online.

After all nodes have rejoined the cluster, the command performs final processing, reconfigures quorum devices, and restores quorum vote counts.

`begin`

Specifies the nodes to assign to the first partition that you upgrade and initiates the dual-partition upgrade process. Run this form of the command from any node of the cluster. Use this upgrade mode after you use the `plan` upgrade mode to determine the possible partition schemes.

First the `begin` upgrade mode records the nodes to assign to each partition. Next, all applications are stopped on one node, then the upgrade mode shuts down the node. The shutdown initiates switchover of each resource group on the node to a node that belongs to the second partition, provided that the node is in the resource-group node list. If the node list of a resource group contains no nodes that belong to the second partition, the resource group remains offline.

The command then repeats this sequence of actions on each remaining node in the first partition, one node at a time.

The nodes in the second partition remain in operation during the upgrade of the first partition. Quorum devices are temporarily unconfigured and quorum vote counts are temporarily changed on the nodes.

`plan`

Queries the cluster storage configuration and displays all possible partition schemes that satisfy the shared-storage requirement. Run this form of the command from any node of the cluster. This is the first command that you run in a dual-partition upgrade.

Dual-partition upgrade requires that each shared storage array must be physically accessed by at least one node in each partition.

The `plan` upgrade mode can return zero, one, or multiple partition solutions. If no solutions are returned, the cluster configuration is not suitable for dual-partition upgrade. Use instead the standard upgrade method.

For any partition solution, you can choose either partition group to be the first partition that you upgrade.

`recover`

Recovers the cluster configuration on a node if a fatal error occurs during dual-partition upgrade processing. Run this form of the command on each node of the cluster.

You must shut down the cluster and boot all nodes into noncluster mode before you run this command.

Once a fatal error occurs, you cannot resume or restart a dual-partition upgrade, even after you run the `recover` upgrade mode.

The `recover` upgrade mode restores the `/etc/vfstab` file and the Cluster Configuration Repository (CCR) database to their original state, before the start of the dual-partition upgrade.

The following list describes in which circumstances to use the `recover` upgrade mode and in which circumstances to take other steps.

- If the failure occurred during `-u begin` processing, run the `-u recover` upgrade mode.
- If the failure occurred after `-u begin` processing completed but before the shutdown warning for the second partition was issued, determine where the error occurred:
  - If the failure occurred on a node in the first partition, run the `-u recover` upgrade mode.
  - If the failure occurred on a node in the second partition, no recovery action is necessary.
- If the failure occurred after the shutdown warning for the second partition was issued but before `-u apply` processing started on the second partition, determine where the error occurred:
  - If the failure occurred on a node in the first partition, run the `-u recover` upgrade mode.
  - If the failure occurred on a node in the second partition, reboot the failed node into noncluster mode.
- If the failure occurred after `-u apply` processing was completed on the second partition but before the upgrade completed, determine where the error occurred:
  - If the failure occurred on a node in the first partition, run the `-u recover` upgrade mode.
  - If the failure occurred on a node in the first partition but the first partition stayed in service, reboot the failed node.

- If the failure occurred on a node in the second partition, run the `-u recover` upgrade mode.

In all cases, you can continue the upgrade manually by using the standard upgrade method, which requires the shutdown of all cluster nodes.

`status`
Displays the status of the dual-partition upgrade. The following are the possible states:

`Upgrade is in progress`
The `scinstall -u begin` command has been run but dual-partition upgrade has not completed.

The cluster also reports this status if a fatal error occurred during the dual-partition upgrade. In this case, the state is not cleared even after recovery procedures are performed and the cluster upgrade is completed by using the standard upgrade method

`Upgrade not in progress`
Either the `scinstall -u begin` command has not yet been issued, or the dual-partition upgrade has completed successfully.

Run the `status` upgrade mode from one node of the cluster. The node can be in either cluster mode or noncluster mode.

The reported state is valid for all nodes of the cluster, regardless of which stage of the dual-partition upgrade the issuing node is in.

The following option is supported with the dual-partition upgrade mode:

`-h` *nodelist*
Specifies a space-delimited list of all nodes that you assign to the first partition. You choose these from output displayed by the `plan` upgrade mode as valid members of a partition in the partition scheme that you use. The remaining nodes in the cluster, which you do not specify to the `begin` upgrade mode, are assigned to the second partition.

This option is only valid with the `begin` upgrade mode.

Examples

Establishing a Two-Node Cluster

The following sequence of commands establishes a typical two-node cluster with Sun Cluster software for Solaris 9 on SPARC based platforms. The example assumes that Sun Cluster software packages are already installed on the nodes.

Insert the installation media on `node1` and issue the following commands:

```
node1# cd media-mnt-pt/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/
node1# ./scinstall -i -F
```

Insert the installation media on `node2` and issue the following commands:

```
node2# cd media-mnt-pt/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/
node2# ./scinstall -i -N node1
```

Establishing a Single-Node Cluster

The following sequence of commands establish a single-node cluster with Sun Cluster software for Solaris 9 on SPARC based platforms, with all defaults accepted. The example assumes that Sun Cluster software packages are already installed on the node.

Insert the installation media and issue the following commands:

```
# cd media-mnt-pt/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/
# ./scinstall -i -F -o
```

Setting Up a Solaris Install Server

The following sequence of commands sets up a JumpStart install server to install and initialize Sun Cluster software for Solaris 9 on SPARC based platforms on a three-node SCI–PCI cluster.

Insert the installation media on the install server and issue the following commands:

```
# cd media-mnt-pt/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/
# ./scinstall -a /export/sc3.1
# cd /export/sc3.1/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/
# ./scinstall –c /export/jumpstart -h node1 -F -A hme2
# ./scinstall –c /export/jumpstart -h node2 -N node1 -A hme2
# ./scinstall –c /export/jumpstart -h node3 -N node1 -A hme2
```

Upgrading the Framework and Data Service Software (Standard or Rolling Upgrade)

The following sequence of commands upgrades the framework and data service software of a cluster to the next Sun Cluster release. This example uses the Sun Cluster version for Solaris 9 on SPARC based platforms. Perform these operations on each cluster node.

Note – For a rolling upgrade, perform these operations on one node at a time, after you use the `clnode evacuate` command to move all resource groups and device groups to the other nodes which will remain in the cluster.

Insert the installation media and issue the following commands:

```
ok> boot -x
# cd media-mnt-pt/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/
# ./scinstall -u update -S interact
# cd /
# eject cdrom

# /usr/cluster/bin/scinstall -u update -s all -d /cdrom/cdrom0
# reboot
```

Performing a Dual-Partition Upgrade

The following sequence of commands uses the dual-partition method to upgrade the framework and data service software of a cluster to the next Sun Cluster release. This examples uses the Sun Cluster version for Solaris 9 on SPARC based platforms. The example queries the cluster for valid partition schemes, assigns nodes to partitions, reboots the node in the first

partition, returns the first partition to operation after upgrade and reboots the node in the second partition, and returns the second partition to the cluster after upgrade.

```
# media-mnt-pt/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/scinstall \
-u plan
  Option 1
    First partition
      phys-schost-1
    Second partition
      phys-schost-2
...
# media-mnt-pt/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/scinstall \
-u begin -h phys-schost-1 phys-schost-3
ok boot -x
```

(Upgrade the node in the first partition)

```
phys-schost-1# /usr/cluster/bin/scinstall -u apply
ok boot -x
```

(Upgrade the node in the second partition)

```
phys-schost-2# /usr/cluster/bin/scinstall -u apply
```

Upgrading the Framework and Data Service Software (Live Upgrade)

The following sequence of commands illustrates the process of performing a live upgrade on an inactive boot environment on a SPARC system that runs Solaris 9. In these commands, `sc31u2` is the current boot environment and `sc32` is the inactive boot environment being upgraded. In this example, the data services that are being upgraded are from the Agents installation media.

Note – The commands shown below typically produce copious output. This output is not shown except where necessary for clarity.

```
# lucreate -c sc31u2 -m /:/dev/dsk/c0t4d0s0:ufs -n sc32
lucreate: Creation of Boot Environment sc32 successful

# luupgrade -u -n sc32 \
-s /net/installmachine/export/solarisX/OS_image
The Solaris upgrade of the boot environment sc32 is complete.

# lumount sc32 /sc32

# cd media-mnt-pt/Solaris_sparc/Product/sun_cluster/Solaris_9/Tools/
# ./scinstall -u update -R /sc32
# cd /usr/cluster/bin
# ./scinstall -R /sc32 -u update -s all -d /cdrom/cdrom0

# cd /
```

```
# eject /cdrom/cdrom0

# luumount -f sc32
# luactivate sc32
Activation of boot environment sc32 successful.
# init 0
ok> boot
```

Uninstalling a Node

The following sequence of commands places the node in noncluster mode, then removes Sun Cluster framework and data-service software and configuration information from the cluster node, renames the global-devices mount point to the default name `/globaldevices`, and performs cleanup. This examples removes a Sun Cluster version for SPARC based platforms.

```
ok> boot -x
# cd /
# /usr/cluster/bin/scinstall -r
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Files

media-mnt-pt`/.cdtoc`

media-mnt-pt`/Solaris_`*arch*`/Product/sun_cluster/.producttoc`

media-mnt-pt`/Solaris_`*arch*`/Product/sun_cluster/Solaris_`*ver*`/ \`
`  Packages/.clustertoc`

media-mnt-pt`/Solaris_`*arch*`/Product/sun_cluster/Solaris_`*ver*`/ \`
`  Packages/.order`

media-mnt-pt`/Solaris_`*arch*`/Product/sun_cluster/Solaris_`*ver*`/ \`
`  Tools/defaults`

media-mnt-pt`/components/`*srvc*`/Solaris_`*ver*`/Packages/.clustertoc`

media-mnt-pt`/components/`*srvc*`/Solaris_`*ver*`/Packages/.order`

`/etc/cluster/ql/cluster_post_halt_apps`

`/etc/cluster/ql/cluster_pre_halt_apps`

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	Java Enterprise System installation media, `SUNWsczu`
Interface Stability	Evolving

See Also Intro(1CL), claccess(1CL), clinterconnect(1CL), clnode(1CL), clsetup(1CL), cluster(1CL), `add_install_client`(1M), `luactivate`(1M), `lucreate`(1M), `lumount`(1M), `luupgrade`(1M), `luumount`(1M), `newfs`(1M), `patchadd`(1M), scconf(1M), scprivipadm(1M), scsetup(1M), scversions(1M), `setup_install_server`(1M), `clustertoc`(4), `netmasks`(4), `networks`(4), `order`(4), `packagetoc`(4), sctransp_dlpi(7p)

Sun Cluster Software Installation Guide for Solaris OS, *Sun Cluster System Administration Guide for Solaris OS*, *Sun Cluster Upgrade Guide for Solaris OS*, *System Administration Guide: IP Services*

Name scnas – manage network-attached storage (NAS) device configuration data for Sun Cluster.

Synopsis

```
scnas [-H]

scnas -a [-H] [-n] -h device-name -t device-type -o specific-options
 [-f input-file]

scnas -c [-H] [-n ] -h device-name -o specific-options
 [-f input-file]

scnas -p [-H] [-h device-name] [-t device-type]

scnas -r [-H ] -h device-name
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scnas` command manages NAS devices in a Sun Cluster configuration. To manage NAS directories in the cluster, use the `scnasdir` command.

You can use the `scnas` command to create the NAS device configuration, to update the NAS type-specific properties, and to remove the device configuration from Sun Cluster. The options to this command are processed in the order in which they are typed on the command line.

The `scnas` command can only be run from an active cluster node. The results of running the command are always the same, regardless of the nodethat is used.

All forms of the `scnas` command accept the `-H` option. Specifying `-H` displays help information. All other options are ignored. Help information is also printed when `scnas` is run without options.

The NAS device must be set up before using the `scnas` command to manage a NAS device. Refer to the documentation for the particular NAS device for procedures for setting up a device.

You can use this command only in the global zone.

Options

Basic Options The following options are common to all forms of the `scnas` command:

`-H`

If this option is specified on the command line at any position, the command prints help information. All other options are ignored and are not executed. Help information is also printed if `scnas` is run with no options.

You can use this option only in the global zone.

`-n`
If this option is specified on the command line at any position, the `scnas` command only checks the usage and does not write the configuration data. If the `-n` option is specified with the `-f` option, the `scnas` command checks the input file for the password.

The following options modify the basic form and function of the `scnas` command. None of these options can be combined on the same command line.

`-a`
Specifies the `add` form of the `scnas` command.

You can use this option only in the global zone.

The `-a` option can be used to add a NAS device into the Sun Cluster configuration.

Depending on the type of your NAS device, you might have to set additional properties. These required properties are also explained in the `-t` option description in the "Additional Options" section.

`-c`
Specifies the `change` form of the `scnas` command. The `-c` option is used to change specific NAS device properties.

You can use this option only in the global zone.

`-r`
Specifies the `remove` form of the `scnas` command. The `-r` option is used to remove the NAS device from the Sun Cluster configuration.

You can use this option only in the global zone.

Before removing a device, all its exported directories must be removed by using `scnasdir`.

`-p`
Specifies the `print` form of the `scnas` command.

You can use this option only in the global zone.

When no other options are given, the `-p` option prints a listing of all the current NAS devices configured in Sun Cluster and all their associated properties. This option can be used with additional options to query a particular device or a particular type of device.

Additional Options

The following additional options can be combined with one or more of the previously described basic options to configure all properties for a device. The device does not need to be online to use these options. Refer to the SYNOPSIS section to see the options that can be used with each form of `scnas`.

The additional options are as follows:

`-h` *device-name*
Use this option to specify the name of the NAS device in the Sun Cluster configuration. The device name identifies the device and can be used to remotely access the device by using `rhs` or `telnet`.

This device name must be specified for the `add`, `change`, and `remove` forms of the `scnas` command.

`-t` *device-type*
The NAS device type. You must specify this option when you add a NAS device to the Sun Cluster configuration. The NAS device type is identified by the vendor name.

You can specify either `sun` for a NAS device from Sun Microsystems, Inc. or `netapp` for a NAS device from Network Appliance, Inc.

Different types of NAS devices have different or in some cases, no properties.

`-o` *specific-options*
Use this option to provide the properties that are specific to a NAS device type. For example, the NAS device from Network Appliance, Inc. has the following property:

`-o userid=`*userid*

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `-f` and `-o` options do *not* apply.

The `userid` property is used by the cluster to perform administrative duties on the device. When you add a `userid` to the device configuration, you are prompted for its password. You can also place the password in a text file and use it by specifying the `-f` option.

`-f`*input-file*
For security reasons, the password cannot be specified in command-line options. To keep the password secure, place it in a text file and specify the file by using the `-f` option. If you do not specify an input file for the password, the command prompts for the password.

Note – You do *not* specify properties for the NAS device from Sun Microsystems, Inc. As this device does not have any properties, the `-f` and `-o` options do *not* apply.

Set permissions of the input file to readable by root and prohibit access by either group or world.

In the input file, the password cannot be entered across multiple lines. Leading white spaces and tabs are ignored. Comments begin with an unquoted pound (#) sign, and continue to the next new line.

The parser ignores all comments. When you use an input file for the device user password, the # sign cannot be part of the password.

Examples

EXAMPLE 1 Adding a NAS Device From Sun Microsystems, Inc. to a Cluster

The following `scnas` command adds a Sun Microsystems, Inc. storage system to the Sun Cluster configuration.

```
# scnas -a -h sunnas1 -t sun
```

EXAMPLE 2 Adding a NAS Device From Network Appliance, Inc. to a Cluster

The following `scnas` command adds a Network Appliance, Inc. storage system to the Sun Cluster configuration.

```
# scnas -a -h netapp1 -t netapp -o userid=root
Please enter password:
```

EXAMPLE 3 Removing a NAS Device From a Cluster

The following `scnas` command removes a NAS device from the Sun Cluster configuration.

```
# scnas -r -h sunnas1
```

Exit Status

The following exit values are returned:

`0`	The command executed successfully.
nonzero	An error has occurred.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Stability	Evolving

See Also

Intro(1CL), clnasdevice(1CL), clquorum(1CL), cluster(1CL), scconf(1M), scnasdir(1M)

Name scnasdir – manage the exported directories on a network-attached storage (NAS) device in a Sun Cluster configuration.

Synopsis

```
scnasdir [-H]

scnasdir [-a] [-H] [-n] -h device-name [-d directory [-d directory...]] [-f input-file]

scnasdir -p [-H] [-h device-name] [-t device-type]

scnasdir -r [-H ] [-n ] -h device-name [-d all | -d directory [-d directory...]]
[-f input-file]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scnasdir` command manages the exported directories on NAS devices in a Sun Cluster configuration. The device must already have been configured in the cluster by using the `scnas` command.

The `scnasdir` command can be used to add directories to a device's cluster configuration, to remove directories from a device's cluster configuration, and to print the directories of a particular device or particular device types.

The options in this command are processed in the order in which they are typed on the command line. The `scnasdir` command can only be run from an active cluster node. The results of running the command are always the same, regardless of the node that is used.

All forms of the `scnasdir` command accept the `-H` option. Specifying `-H` displays help information, and all other options are ignored and not executed. Help information is also printed when `scnasdir` is run without options.

You can use this command only in the global zone.

Options

Basic Options The following options are common to all forms of the `scnasdir` command:

`-H`

If this option is specified on the command line at any position, the command prints help information. All other options are ignored and are not executed. Help information is also printed if `scnasdir` is run with no options.

You can use this option only in the global zone.

`-n`

If this option is specified on the command line at any position, the `scnasdir` command only checks the usage and does not write the configuration data. If the `-n` option is specified with the `-f` option, the `scnasdir` command displays the data that will be processed for the

user to review.

The following options modify the basic form and function of the `scnasdir` command. None of these options can be combined on the same command line.

`-a`
Specifies the add form of the `scnasdir` command. The `-a` option can be used to add directories into the device's Sun Cluster configuration.

You can use this option only in the global zone.

`-p`
Specifies the print form of the `scnasdir` command. When no other option is given, this `-p` option prints a listing of all the directories of all the NAS devices configured in Sun Cluster. This option can be used with additional options to query a particular device or particular types of NAS devices.

You can use this option only in the global zone.

`-r`
Specifies the remove form of the `scnasdir` command. The `-r` option is used to remove all the directories, or the specified directories of a NAS device from its Sun Cluster configuration.

You can use this option only in the global zone.

Additional Options

The following additional options can be combined with one or more of the previously described basic options to manage the directories of a device.

The additional options are as follows:

`-h` *device-name*
Use this option to specify the name of the NAS device in the Sun Cluster configuration. The `-h` option identifies the device and can be used to remotely access the device by using `rhs` or `telnet`.

This device name must be specified for the add, change, and remove forms of the `scnasdir` command.

`-d` *all* | *directory*
Use this option to list the directories (or volumes) exported on the NAS device to be configured into the Sun Cluster. These directories must be created and exported on the device before using the `scnasdir` command. See the documentation for the NAS device type for procedures for exporting directories.

The `-d all` option can only be accepted by the remove option, `-r`.

The directories must be specified by using either the `-d` option, or the `-f` option, for the add and remove forms of the `scnasdir` command.

`-f` *input-file*
Directories can be placed into a plain text file, one directory per line, and used with the `-f` option. Leading white spaces and tabs are ignored. Comments begin with an unquoted pound (#) sign, and continue to the next new line. The parser ignores all comments.

Examples

EXAMPLE 1 Adding Two NAS Storage Device Directories to a Cluster

The following `scnasdir` command adds two directories of a NAS device to the Sun Cluster configuration.

```
# scnasdir -a -h sunnas1 -d /vol/DB1 -d /vol/DB2
```

EXAMPLE 2 Removing All of a NAS Storage Device's Directories From a Cluster

The following `scnasdir` command removes all the directories that are configured for a NAS device.

```
# scnasdir -r -h sunnas1 -d all
```

Exit Status

The following exit values are returned:

`0`	The command executed successfully.
nonzero	An error has occurred.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), clnasdevice(1CL), clquorum(1CL), cluster(1CL), scconf(1M), scnas(1M)

Name scprivipadm – administer the private IP address range

Synopsis

```
scprivipadm -c netaddr=netaddr[,netmask=netmask]

scprivipadm -c netaddr=netaddr[,maxnodes=nodes,maxprivatenets=privnets]

scprivipadm -c netaddr=netaddr[,netmask=netmask,maxnodes=nodes,maxprivatenets=privnets]

scprivipadm -p

scprivipadm -R
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scprivipadm` command modifies the current IP address range that is assigned to the Sun Cluster private interconnect.

All nodes in the cluster must be in noncluster mode before you run any form of this command. Run this command from one node in the cluster.

The `scprivipadm` command takes as input the private network address. Optionally, the command also takes one or both of the following:

- The netmask
- The maximum number of nodes and the maximum number of private networks that are ever expected to be in the cluster

The command then performs the IP address assignment for the physical adapters and the per-node IP addresses.

You can use this command only in the global zone.

Options The following options are supported:

`-c` Modifies the IP address range that is currently assigned to the cluster. Run the `-c` option on each node of the cluster.

You can use this option only in the global zone.

The `-c` option supports the following suboptions:

`netaddr=`*netaddr*	Specifies the private network address
`netmask=`*netmask*	Specifies the netmask
`maxnodes=`*nodes*	Specifies the maximum expected number of nodes in the cluster

`maxprivatenets=`*privnets* Specifies the maximum expected number of private networks in the cluster

The `-c` option performs the following tasks for each combination of suboptions:

- If you specify the `netaddr` suboption alone, the command assigns the default netmask, `255.255.248.0`, to the private interconnect. The default IP address range accommodates a maximum of 64 nodes and 10 private networks.
- If you also specify the `netmask` suboption, the value that you specify must be equal to or greater than the default netmask. If the specified netmask is less than the default netmask, the command fails and exits with an error. If the specified netmask is equal to or greater than the default netmask, the command assigns the specified netmask to the private interconnect. The resulting IP address range can accommodate a maximum of 64 nodes and 10 private networks.

  To assign a smaller IP address range than the default, specify the `maxnodes` and `maxprivatenets` suboptions.
- If you also specify the `maxnodes` and `maxprivatenets` suboptions, the command calculates the minimum netmask to support the specified number of nodes and private networks. The command then assigns the calculated netmask to the private interconnect. The maximum value for *nodes* is 64 and the minimum value is 2. The maximum value for *privnets* is 128 and the minimum value is 2.
- If you also specify the `netmask` suboption as well as the `maxnodes` and `maxprivatenets` suboptions, the command calculates the minimum netmask that supports the specified number of nodes and private networks. The command compares that calculation to the specified netmask. If the specified netmask is less than the calculated netmask, the command fails and exits with an error. If the specified netmask is equal to or greater than the calculated netmask, the command assigns the specified netmask to the private interconnect. The maximum value for *nodes* is 64 and the minimum value is 2. The maximum value for *privnets* is 128 and the minimum value is 2.

If the `-c` option fails, you must run the `-R` option on each node to repair the configuration before you rerun the `-c` option.

Users other than superuser require `solaris.cluster.modify` Role-Based Access Control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

`-R` Repairs the cluster configuration. Use this option if the command fails while modifying the IP address range on the cluster nodes and the failure results in inconsistent cluster configuration on the nodes.

You can use this option only in the global zone.

Run the `-R` option on each node of the cluster.

The `-R` option repairs the cluster configuration and removes any inconsistencies that were caused by a failure to modify the IP address range on all nodes.

If you attempt to rerun the `-c` option without first running the `-R` option, the configuration change might again fail.

Users other than superuser require `solaris.cluster.modify` Role-Based Access Control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

`-p` Displays the current private network address that is assigned to the private interconnect. Run the `-p` option from any node.

You can use this option only in the global zone.

The `-p` option prints the following information:

- The private network address
- The IP address range in the form of a netmask
- The maximum number of nodes and the maximum number of private networks that can be supported by the IP address range

Users other than superuser require `solaris.cluster.read` Role-Based Access Control (RBAC) authorization to use this subcommand. See the `rbac`(5) man page.

To display the current private network address from a node that is in cluster mode, instead run the `scconf -p` command or the `cluster show-netprops` command.

Examples

EXAMPLE 1 Calculating a Custom Private IP Address Range

The following command specifies the private network address `172.16.0.0` and calculates the netmask. The command specifies that the calculated netmask must support up to sixteen nodes and up to four private networks in the cluster.

```
# scprivipadm -c netaddr=172.16.0.0,maxnodes=16,maxprivatenets=4
```

EXAMPLE 2 Specifying a Private Network Address and Netmask

The following command specifies the private network address `172.16.0.0` and the netmask `255.255.248.0`.

EXAMPLE 2 Specifying a Private Network Address and Netmask *(Continued)*

```
# scprivipadm -c netaddr=172.16.0.0,netmask=255.255.248.0
```

Exit Status The `scprivipadm` command returns with a non-zero value if either of the following conditions occur:

- Invalid arguments were provided.
- The command was unable to successfully modify the IP address range on all nodes of the cluster.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscu`
Interface Stability	Evolving

See Also Intro(1CL), cluster(1CL), scconf(1M), scinstall(1M), `netmasks`(4), `networks`(4), `rbac`(5)

Sun Cluster Software Installation Guide for Solaris OS, *Sun Cluster System Administration Guide for Solaris OS*, *System Administration Guide: IP Services*

Notes The superuser can run all forms of this command. Users other than superuser require RBAC authorizations. See the following table.

Option	RBAC Authorization
`-c`	`solaris.cluster.modify`
`-R`	`solaris.cluster.modify`
`-p`	`solaris.cluster.read`

Name scprivipd – Sun Cluster Private IP address service daemon

Synopsis

```
/usr/cluster/lib/sc/scprivipd
```

Description Sun Cluster private IP addresses can be assigned to non-global zones by using the `scconf` command or the `clnode` command. The `scprivipd` daemon is a server daemon that is used for the management of these private IP addresses that are assigned to non-global zones. The `scprivipd` daemon is started at system boot time. It is used to configure or unconfigure the private IP addresses that are assigned to non-global zones on zone boot or shutdown or as a result of `scconf` operations.

Diagnostics The `scprivipd` daemon has no direct `stdin`, `stdout`, or `stderr` connection to the outside. All diagnostic messages are logged through the `syslog` function.

Notes The `scprivipd` daemon must be run in superuser mode.

The `scprivipd` daemon is a Service Management Facility (SMF) service and is started through SMF. Also, if the `scprivipd` daemon is killed by a signal, it is automatically restarted by SMF.

The `SIGTERM` signal can be used to kill `scprivipd` gracefully. Other signals should not be used to kill the daemon.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Private

See Also clnode(1CL), scconf(1M), `syslog`(3C), `attributes`(5)

Name scqsd – quorum server daemon

Synopsis
```
/usr/cluster/lib/sc/scqsd  [-i instance] [-p port-number]
 [-d quorum-directory]
```

Description The `scqsd` daemon starts automatically when the node is booted. Installing the `SUNWscqsr` package sets up a startup file, `/etc/scqsd/scqsd.conf`, which contains information for starting a single quorum server using default values. See the scqsd.conf(4) man page for the format of this configuration file. Installing the package also adds an entry to the `/etc/services` file. This entry specifies the default port number to be used by the quorum server. If you need multiple quorum servers on the same machine, then create one entry per instance in the `/etc/scqsd/scqsd.conf` file.

You specify options such as port number and quorum directory in the configuration file. To start multiple quorum servers, you must provide at least a unique port number and a unique quorum directory for each instance of the quorum server.

The use of a startup file enables administrators to start and stop different instances of the quorum server through a single script. For details about starting and stopping the daemon on the command line, see the clquorumserver(1CL) man page.

Examples **EXAMPLE 1** Starting a Quorum Server

The following command starts a quorum server listening on port 2000.

```
# clquorumserver start 2000
```

EXAMPLE 2 Stopping All Quorum Servers

The following command stops all configured quorum servers.

```
# clquorumserver stop
```

Exit Status The following exit values are returned:

0	The command completed successfully.
nonzero	An error occurred.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscqs`
Stability	Deprecated

See Also Intro(1CL), clquorumserver(1CL), scconf_quorum_dev_quorum_server(1M), scqsd.conf(4)

Name scrgadm – manage registration and unregistration of resource types, resource groups, and resources

Synopsis

```
scrgadm -p[v[v]] [-t resource_type_name] [-g resource_group_name] [-j resource_name]

scrgadm -a -t resource_type_name [-h RT_installed_node_list] [-f registration_file_path]

scrgadm -a -g RG_name [-h node-zone-list] [-y property=value...]

scrgadm -a -j resource_name -t resource_type_name -g RG_name [-y property=value...]
[-x "extension_property[{node_specifier}]=value..."]

scrgadm -a -L -g RG_name -l hostnamelist [-j resource_name] [-n netiflist]
[-y property=value...]

scrgadm -a -S -g RG_name -l hostnamelist [-j resource_name] [-n netiflist]
[-X auxnodelist] [-y property=value...]

scrgadm -c -t resource_type_name [-h RT_installed_node_list] [-y RT_system={TRUE|FALSE}]

scrgadm -c -g RG_name [-h node-zone-list] -y property=value...

scrgadm -c -j resource_name [-y property...]
[-x "extension_property[{node_specifier}]=value..."]

scrgadm -r -t resource_type_name

scrgadm -r -g RG_name

scrgadm -r -j resource_name
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

A resource type specifies common properties and callback methods for all resources of that type. Before you can create a resource of a particular type, you must first register the resource type using the following form of the command:

```
# scrgadm -a -t resource_type_name
```

A resource group contains a set of resources, all of which are brought online or offline together on a given node or set of nodes. You first create an empty resource group before placing any resources in it. To create a resource group, use the following command:

```
# scrgadm -a -g RG_name
```

There are two types of resource groups: failover and scalable.

A failover resource group is online on only one node at a time. A failover resource group can contain resources of any type although scalable resources that are configured in a failover resource group run on only one node at a time.

To create a failover resource group named `MyDatabaseRG`, use the following command:

```
# scrgadm -a -g MyDatabaseRG
```

A scalable resource group can be online on several nodes at once. A scalable resource group can contain only resources that support scaling and cannot contain resources that are constrained, by their resource type definition, to only failover behavior.

To create a scalable resource group named `MyWebServerRG`, use the following command:

```
# scrgadm -a -g MyWebServerRG \
-y Maximum_primaries=integer \
-y Desired_primaries=integer
```

A newly created resource group is in an `UNMANAGED` state. After creating resources in the group, use the `scswitch` command to put a resource group in a `MANAGED` state.

To create a resource of a given type in a resource group, use the following command:

```
# scrgadm -a -j resource_name -t resource_type_name -g RG_name
```

Creating a resource causes the underlying RGM mechanism to take several actions. The underlying RGM mechanism calls the `VALIDATE` method on the resource to verify that the property settings of the resource are valid. If the `VALIDATE` method completes successfully and the resource group has been put in a `MANAGED` state, the RGM initializes the resource by calling the `INIT` method on the resource. The RGM then brings the resource online if it is enabled and its resource group is online.

To remove a managed resource group, first remove all resources from that resource group. To remove a resource, first disable it with the `scswitch` command. Removing a resource causes the RGM to clean up after the resource by calling the `FINI` method on that resource.

You can use some forms of this command in a non-global zone, referred to simply as a zone. For more information about valid uses of this command in zones, see the descriptions of the individual options. For ease of administration, use this command in the global zone.

Options

Action Options

Action options specify the actions performed by the command. Only one action option is allowed on the command line.

The following action options are supported:

`-a`

Adds a new configuration. Use with these options:

`-g` Creates a resource group.

You can use this option only in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-j` Creates a resource.

You can use this option in a non-global zone if the following conditions are met:

- The non-global zone exists in the node list of the specified resource group.
- The `Global_zone` property of the resource type is set to `False`.

Otherwise, you can use this option in only the global zone. For ease of administration, use this form of the command in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-t` Adds a resource type.

You can use this option only in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-c`

Modifies an existing configuration. Only values of the specified properties are set. Other properties retain their current values. Use with these options:

`-g` Modifies a resource group.

You can use this option in a non-global zone if the zone exists in the resource group's node list, unless the command modifies the node list. Otherwise, you can use this option only in the global zone. For ease of administration, use this form of the command in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-j` Modifies a resource.

If you use this option in a non-global zone, this option successfully operates only on resources that can be mastered by that zone. If you use this option in the global zone, this option can operate on any resource. For ease of administration, use this form of the command in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-t` Modifies a resource type.

You can use this option only in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-r`
Removes configuration. Use with these options:

`-g` Removes a resource group.

You can use this option only in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-j` Removes a resource.

You can use this option in a non-global zone if the following conditions are met:

- The non-global zone exists in the node list of the specified resource group.
- The `Global_zone` property of the resource type is set to `False`.

Otherwise, you can use this option in only the global zone. For ease of administration, use this form of the command in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-t` Removes a resource type.

You can use this option only in the global zone.

You need `solaris.cluster.resource.modify` RBAC authorization to use this command option with `-a`, `-c`, or `-r`. See `rbac`(5).

`-p`
Displays existing configuration information.

You can use this option in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone. Use with these options:

`-g` *resource_group_name*
Displays specific resource group configuration information.

You need `solaris.cluster.resource.read` RBAC authorization to use this command option with `-p`. See `rbac`(5).

`-j` *resource_name*
Displays specific resource configuration information.

You need `solaris.cluster.resource.read` RBAC authorization to use this command option with `-p`. See `rbac`(5).

`-t` *resource_type_name*
Displays specific resource type configuration information.

You need `solaris.cluster.resource.read` RBAC authorization to use this command option with `-p`. See `rbac`(5).

`-v`[`v`]
Displays more verbose output.

You need `solaris.cluster.resource.read` RBAC authorization to use this command option with `-p`. See `rbac`(5).

If you do not specify any `-g`, `-j`, or `-t` options, information about all resource types, resource groups, and resources that are currently configured on the cluster are provided by default.

Multiple `-g`, `-j`, and `-t` options are supported and can be combined with any combination of `-v` options.

You can use up to two `-v` options on a single command line.

Target Options

Target options identify the target object. The following target options are supported:

Note – Property names for resource groups, resources, and resource types are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify property names.

`-g` *RG_name*
Resource group.

`-j` *resource_name*
Resource. When used with the `-a` option, the `-t` and `-g` target options must be specified in the command to indicate the type of the resource that is to be instantiated and the name of the containing resource group.

`-t` *resource_type_name*
Resource type.

Resource Type-Specific Options

The following options are supported:

`-f` *registration_file_path*
Is valid with `-a`. Specifies the path name of the resource type registration (RTR) file. By convention, the RTR file resides in the `/opt/cluster/lib/rgm/rtreg` directory. If the RTR file is not located in this directory, you must specify this option.

`-h` *RT_installed_node_list*
Is valid with `-a` and `-c`. Specifies a comma-separated list of node or zone names upon which this resource type is installed. Resources of this type can be instantiated only in resource groups whose node list is a subset of this list.

The `-h` option is optional with the `-a` option. If `-h` is not specified, it implies that the resource type has been installed on all nodes. Doing so permits resources of this type to be instantiated in any resource group.

When used with the `-c` option, `-h` must be specified with either a new installed node list or with an escaped wildcard character (`\*`). The wildcard character indicates that the resource type has been installed on all nodes and zones.

Note – A comma is not allowed in a node name.

`-t` *resource_type_name*
Is valid with `-a`, `-c`, and `-r`. A resource type is defined by a resource type registration file that specifies standard and extension property values for the resource type. Placing a valid resource type registration file in the well-known directory where registration files are usually installed (`/opt/cluster/lib/rgm/rtreg`) enables the shorthand notation:

```
# scrgadm -a -t SUNW.rt:2.0
```

As a result, you do not need to use the following notation:

```
# scrgadm -a -t rtn -f full_path_to_SUNW.rt:2.0
```

To view the names of the currently registered resource types, use the following command:

```
# scrgadm -p
```

Starting in Sun Cluster 3.1, the syntax of a resource type name is as follows:

vendor_id.*resource_type*:*version*

The three components of the resource type name are properties specified in the RTR file as *Vendor_id*, *Resource_type*, and *RT_version*. The `scrgadm` command inserts the period and colon delimiters. The optional *Vendor_id* prefix is necessary only if it is required to distinguish between two registration files of the same name provided by different vendors. The *RT_version* is used for upgrading from one version of a data service to another version of the data service.

To ensure that the *Vendor_id* is unique, use the stock symbol for the company that is creating the resource type. The *resource_type_name* that is used with the `-t` option can either be the full resource type name or an abbreviation that omits the *Vendor_id*. For example, both `-t SUNW.iws` and `-t iws` are valid. If there are two resource types in the cluster with names that differ only in the *Vendor_id* prefix, the use of the abbreviated name fails.

The `scrgadm` command fails to register the resource type if the *RT_version* string includes a blank, tab, slash (`/`), backslash (`\`), asterisk (`*`), question mark (`?`), left square bracket (`[`), or right square bracket (`]`) character.

When you specify the *resource_type_name* with the `-t` option, you can omit the version component if only one version is registered.

Resource type names that you created before the Sun Cluster 3.1 release continue to conform to the following syntax:

vendor_id`.`*resource_type*

`-y RT_system={TRUE|FALSE}`
Sets the `RT_system` property of a resource type either to `TRUE` or to `FALSE`. The default value of the `RT_system` property is `FALSE`. See rt_properties(5) for a description of the `RT_system` property.

Resource Group-Specific Options

The following options are supported:

`-h` *node-zone-list*
Is valid with `-a` and `-c`. This option is a shortcut for –y `Nodelist=`*node-zone-list*.

To specify a non-global zone, use the following syntax:

node`:`*zone*

The *node* component is the name of the physical node where *zone* is located. The *zone* component is the name of the zone that you want to include in `Nodelist`. For example, to specify the non-global zone `zone-1` which is located on the node `phys-schost-1`, you specify the following text:

```
phys-schost-1:zone-1
```

`-y` *property=value*
Is valid with `-a` and `-c`. Multiple instances of `-y` *property=value* are allowed. The form of the *value* is dictated by each *property*. In the following example, *property1* takes a single string as the *value*, while *property2* takes a comma-separated string array:

`-y` *property1=value1* `-y` *property2=value2a*`,`*value2b*

To set a string property to an empty value, use this option without specifying a value, as follows:

`-y` *property=*

Recognition of `-y` property names is not case-sensitive.

See rg_properties(5) for a description of the resource group properties.

Resource-Specific Options

The following options are supported:

`-x` *extension_property=value*
`-x` “*extension_property*{*node_specifier*}=*value*”

Is valid with `-a` and `-c`. Multiple instances of `-x` *extension_property=value* or `-x` "*extension_property*{*node_specifier*}=*value*" are allowed.

node_specifier is an *optional* qualifier that indicates that the value of *extension_property* is to be set or changed on *only* the specified node or nodes or zone or zones. The value for the specified property is not set or changed on other nodes or zones in the cluster. If you do not include *node_specifier*, the value for the specified property is set or changed on all nodes and zones in the cluster. Examples of the syntax of *node_specifier* are as follows:

```
-x "myprop{phys-schost-1}=100"
```

You specify the braces ({ }) to indicate the particular node or nodes or zone or zones on which you want to set the property.

You can also use the following syntax for *node_specifier* to specify different values on two different nodes at the same time:

```
-x "myprop{phys-schost-1}=100" -x "myprop{phys-schost-2}=10"
```

Alternately, you can use the following syntax to set or change one value on two different nodes at the same time:

```
-x "myprop{phys-schost-1,phys-schost-2}=100"
```

The form of *value* is dictated by each *extension_property*. In the following example, *extension_property1* takes a single string as the *value*, while *extension_property2* takes a comma-separated string:

```
-x "extension_property1{node_specifier}=value1" \
-x "extension_property2{node_specifier}=value2a,value2b"
```

For information about the extension properties that are available for a particular data service, see the man page for that data service.

`-y` *property=value*

Is valid with `-a` and `-c`. Multiple instances of `-y` *property=value* are allowed. The form of the *value* is dictated by each *property*. In the following example, *property1* takes a single string as the *value*, while *property2* takes a comma-separated string array:

```
-y property1=value1 -y property2=value2a,value2b
```

To set a property to an empty value, use this option without specifying a value, as follows:

```
-y property=
```

Recognition of `-y` *property* names is not case-sensitive.

See the r_properties(5) man page for a description of the resource properties.

`LogicalHostname` Specific Options

These options apply to logical host name resources. There are no special commands for removing a `LogicalHostname` resource:

```
# scrgadm -r -j resource_name
```

resource_name is the same name that is supplied with the optional `-j` option when you create the `LogicalHostname` resource. If the `-j` option and *resource_name* are omitted when the `LogicalHostname` resource is created, then the name is generated by `scrgadm`.

The following options are supported:

`-j` *resource_name*
The `-j` option is required when you use an IP address rather than a host name as the first argument to the `-l` *hostnamelist* option.

Use `-j` with `-a` to explicitly name a `LogicalHostname` resource when the resource is created and with `-r` to remove a resource from a resource group. If you do not use the `-j` option to explicitly name the `resource`, the `scrgadm` command creates the resource and assigns the name of the first host name in *hostnamelist* to that resource.

`-L`
Indicates that the options that are used on the command line apply to a logical host name. If you issue the command when any cluster node is not an active cluster member, you must also use the `-n` *netiflist* option.

`-l` *hostnamelist*
Specifies the IPv4 or IPv6 addresses to be shared. Use host names even though you can specify IP addresses. *hostnamelist* is a comma-separated list of host names that are to be made available by this `LogicalHostname` resource.

`-n` *netiflist*
Specifies the list of network interfaces. The `-L` option requires the `-n` option if the command is issued when any cluster node is not an active cluster member.

The *netiflist* takes the following form:

netif`@`*node*`[,...]`

netif may be given as network adapter name, such as `le0`, or as an IP Network Multipathing group name, such as `sc_ipmp`. The *node* may be a node name or node identifier. All nodes in the *nodelist* of the resource group must be represented in *netiflist*. If `-n` *netiflist* is omitted, an attempt is made to discover a net adapter on the subnet identified by the *hostnamelist* for each node in the *nodelist*. Single-adapter IP Network Multipathing groups are created for discovered network adapters not already in an IP Network Multipathing group. Similarly, a single-adapter IP Network Multipathing group is created for a named adapter, if a group does not already exist.

Refer to the NOTES section for more information.

`-y` *property=value*
Refer to the Resource-Specific Options section for details.

SharedAddress Specific Options

All of the `LogicalHostname`-specific options also apply to `SharedAddress` resources with the following changes and additions:

`-S`
Indicates that the options that are used on the command line apply to a shared address.

`-X` *auxnodelist*
Specifies a comma-separated list of node names or identifiers. Entries on this list must be members of the cluster. These nodes are nodes that may host the specified shared addresses, but never serve as the primary node in the case of failover.

This list is mutually exclusive with *nodelist*. See the description of *nodelist* under Resource Group-Specific Options.

Exit Status

The following exit values are returned:

`0` The command completed successfully.

A warning message might be written to the standard error even when this command completes successfully.

nonzero An error has occurred.

Writes an error message to standard error when it exits with nonzero status.

Some operations are not permitted on resource types whose `RT_System` property is `TRUE`. Similarly, some operations are not permitted on a resource group (and its resources) whose `RG_System` property is `TRUE`. See rt_properties(5) and rg_properties(5).

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Interface Stability	Evolving

See Also

Intro(1CL), clreslogicalhostname(1CL), clresource(1CL), clresourcegroup(1CL), clresourcetype(1CL), clressharedaddress(1CL), `ifconfig`(1M), scstat(1M), scswitch(1M), r_properties(5), `rbac`(5), rg_properties(5), rt_properties(5)

Notes

A network adapter that is not already configured for use cannot be discovered or placed into an IP Network Multipathing group during `LogicalHostname` and `SharedAddress` add operations. See `ifconfig`(1M).

If `scrgadm` exits nonzero with the error message `cluster is reconfiguring`, the requested operation might have completed successfully, despite the error status. If you doubt the result, you can execute `scrgadm` again with the same arguments after the reconfiguration is complete.

Name scsetup – interactive cluster configuration tool

Synopsis

```
scsetup [-f logfile]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scsetup` command provides the following configuration capabilities, depending on what state the cluster is in when you issue the command:

- When you run the `scsetup` command at post-installation time, the command performs initial setup tasks, such as configuring quorum devices and resetting the `installmode` property. If you did not use automatic quorum configuration when you created the cluster, run the `scsetup` command immediately after the cluster is installed. Ensure that all nodes have joined the cluster before you run the `scsetup` command and reset the `installmode` property.

  If you used automatic quorum configuration when you created the cluster, you do not need to run the `scsetup` command after cluster installation. The automatic quorum configuration feature also resets the `installmode` property of the cluster.
- When you run the command during normal cluster operation, the `scsetup` command provides a menu-driven utility. You can use this utility to perform most ongoing cluster-administration tasks.
- When you issue the command from a node that is in noncluster mode, the `scsetup` utility provides a menu-driven utility for changing and displaying the private IP address range. You must reboot all nodes into noncluster mode before you start this form of the `scsetup` utility.

You can issue the `scsetup` command from any node in the cluster.

You can use this command only in the global zone.

Options The following options are supported:

`-f` *logfile* Specifies the name of a log file to which commands can be logged. If you specify this option, most command sets that the `scsetup` utility generates are run and logged, or only logged, depending on user responses.

Attributes See `attributes`(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also Intro(1CL), cltelemetryattribute(1CL), cldevicegroup(1CL), clnode(1CL), clquorum(1CL), clreslogicalhostname(1CL), clresourcegroup(1CL), clresourcetype(1CL), clressharedaddress(1CL), cluster(1CL),

Name scshutdown – shut down a cluster

Synopsis

```
scshutdown [-y] [-g grace-period] [message]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scshutdown` utility shuts down an entire cluster in an orderly fashion.

Before starting the shutdown, `scshutdown` sends a warning message, then a final message asking for confirmation.

Only run the `scshutdown` command from one node.

The `scshutdown` command performs the following actions when it shuts down a cluster:

- Changes all functioning resource groups on the cluster to an offline state. If any transitions fail, `scshutdown` does not complete and displays an error message.
- Unmounts all cluster file systems. If any unmounts fail, `scshutdown` does not complete and displays an error message.
- Shuts down all active device services. If any transition of a device fails, `scshutdown` does not complete and displays an error message.
- Runs `/usr/sbin/init 0` on all nodes. See `init`(1M) for more information.

You can use this command only in the global zone.

You need `solaris.cluster.system.admin` RBAC authorization to use this command. See `rbac`(5).

Options The following options are supported:

`-g` *grace-period*	Changes the number of seconds from the 60-second default to the time specified by *grace-period*.
`-y`	Pre-answers the confirmation question so the command can be run without user intervention.

Operands The following operands are supported:

message	Is a string that is issued after the standard warning message `The system will be shut down in ...` is issued. If *message* contains more than one word, delimit it with single (') or double (") quotation marks. The warning message and the user-provided *message* are output when there are `7200`, `3600`, `1800`, `1200`, `600`, `300`, `120`, `60`, and `30` seconds remaining before `scshutdown` begins.

Examples

EXAMPLE 1 Shutting Down a Cluster

```
phys-palindrome-1# scshutdown
```

Exit Status

The following exit values are returned:

`0`	The command completed successfully.
nonzero	An error occurred. Error messages are displayed on the standard output.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cluster(1CL), `shutdown`(1M), `init`(1M), `attributes`(5)

Name scsnapshot – retrieve configuration data about resource groups, resource types, and resources, and generate a shell script

Synopsis

```
scsnapshot [-s scriptfile] [-o imagefile]

scsnapshot [-s scriptfile] oldimage newimage
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scsnapshot` tool retrieves information from the Cluster Configuration Repository (CCR) about configuration data that is related to resource groups, resource types, and resources. The `scsnapshot` tool formats the configuration data as a shell script that can be used for the following purposes:

- To replicate configuration data on a cluster that has no configured resource groups, resource types, and resources
- To upgrade configuration data on a cluster that has configured resource groups, resource types, and resources

The `scsnapshot` tool retrieves configuration data only from the CCR. Other configuration data is ignored. The `scsnapshot` tool does not take into account the dynamic state of different resource groups, resource types, and resources.

You can use this command only in the global zone.

Options The following options are supported by the `scsnapshot` tool. If you use an incorrect command option, the correct way to use the command option is displayed.

`-s` *scriptfile*
Stores the generated script in a file called *scriptfile*.

You can use this option only in the global zone.

If this option is not specified, the generated script is written to the standard output.

If a file called *scriptfile* already exists, it is renamed as *scriptfile*`.old`, and a new file called *scriptfile* is created. If a file called *scriptfile*`.old` already exists, it is overwritten.

`-o` *imagefile*
Stores the generated image file in a file called *imagefile*.

You can use this option only in the global zone.

If this option is not specified, an image file is not generated.

If a file called *imagefile* already exists, it is renamed as *imagefile*`.old`, and a new file called *imagefile* is created. If a file called *imagefile*`.old` already exists, it is overwritten.

oldimage
Specifies an image file that contains the old configuration data.

newimage
Specifies an image file that contains the new configuration data.

Extended Description

The output of the `scsnapshot` tool is an executable Bourne shell-based script. Before you run the script, you might need to manually change some properties to reflect the features of your host.

The script compares the following characteristics of the local cluster to the cluster where the script was generated:

- Machine architecture
- Version of the Solaris Operating System
- Version of the Sun Cluster software

If the characteristics are not the same, the script writes an error and ends. A message asks whether you want to rerun the script by using the `-f` option. The `-f` option forces the script to run, despite any difference in characteristics.

The script generated by the `scsnapshot` tool verifies that the Sun Cluster resource type exists on the local cluster. If the resource type does not exist on the local cluster, the script writes an error and ends. A message asks whether you want to install the missing resource type before you run the script again.

To run a script that is generated by the `scsnapshot` tool, you need `solaris.cluster.resource.modify` RBAC authorization. For more information, see the `rbac`(5) man page.

Usage

This section describes how you can use the `scsnapshot` tool.

Retrieving Configuration Data for Resource Groups, Resource Types, and Resources

`scsnapshot [-s` *scriptfile*`] [-o` *imagefile*`]`

Used without the `-o` option, the `scsnapshot` tool generates a script that creates configuration data for clusters that do not already have configured resource groups, resource types, and resources. See Example 1.

Used with the `-o` option, the `scsnapshot` tool produces an image file that represents the configuration data. The image file can be used in further invocations of the `scsnapshot` tool to upgrade configuration data on a cluster. See Example 2.

To use the `scsnapshot` tool to retrieve configuration data, you need `solaris.cluster.resource.read` role-based access control (RBAC) authorization. For more information, see the `rbac`(5) man page.

To track differences between versions of configuration data, store the image files in a source control system such as SCCS.

Upgrading Configuration Data for Resource Groups, Resource Types, and Resources

`scsnapshot [-s` *scriptfile*`]` *oldimage newimage*

The `scsnapshot` tool generates a shell script that can be used to upgrade the configuration data that is contained in the *oldimage* file with the configuration data that is contained in the *newimage* file.

To use the `scsnapshot` tool to upgrade configuration data, you do not need specific RBAC authorization.

Examples

EXAMPLE 1 To Generate a Shell Script That Retrieves Configuration Data for Resources Groups, Resource Types, and Resources

The script that is generated in this example is called `scriptfile.sh`.

```
example% scsnapshot -s scriptfile.sh
```

EXAMPLE 2 To Generate a Shell Script That Retrieves Configuration Data and Stores an Image File

The script that is generated in this example is called `scriptfile.sh`. The configuration data is stored in an image file called `imagefile`.

```
example% scsnapshot -s scriptfile.sh -o imagefile
```

EXAMPLE 3 To Generate a Shell Script That Upgrades Configuration Data on One Cluster With Configuration Data From Another Cluster

This example creates a script that upgrades the configuration data on `cluster1` to match the configuration data on `cluster2`. The configuration data for `cluster1` is in a file called `imagefile1`, and the configuration data for `cluster2` is in a file called `imagefile2`. The name of a shell script is not specified, so the generated script is written to the standard output.

```
example% scsnapshot imagefile1 imagefile2
```

Exit Status

The following exit status codes are returned:

`0`	The command completed successfully.
nonzero	An error occurred.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), `attributes`(5), clconfiguration(5CL), `rbac`(5), rg_properties(5), rt_properties(5), r_properties(5)

Name scstat – monitor the status of a Sun Cluster configuration

Synopsis

```
scstat [-DWginpqv [v]] [-h node]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scstat` command displays the current state of Sun Cluster components. Only one instance of the `scstat` command needs to run on any machine in the Sun Cluster configuration.

When run without any options, `scstat` displays the status for all components of the cluster. This display includes the following information:

- A list of cluster members
- The status of each cluster member
- The status of resource groups and resources
- The status of every path on the cluster interconnect
- The status of every disk device group
- The status of every quorum device
- The status of every IP network multipathing (IPMP) group and public network adapter

From a non-global zone, referred to simply as a zone, you can run all forms of this command except the `-i` option. When you run the `scstat` command from a non-global zone, the output is the same as when run from the global zone except that no status information is displayed for IP network multipathing groups or public network adapters.

You need `solaris.cluster.device.read`, `solaris.cluster.transport.read`, `solaris.cluster.resource.read`, `solaris.cluster.node.read`, `solaris.cluster.quorum.read`, and `solaris.cluster.system.read` RBAC authorization to use this command without options. See `rbac`(5).

Resources and Resource Groups

The resource state, resource group state, and resource status are all maintained on a per-node basis. For example, a given resource has a distinct state on each cluster node and a distinct status on each cluster node.

The resource state is set by the Resource Group Manager (RGM) on each node, based only on which methods have been invoked on the resource. For example, after the `STOP` method has run successfully on a resource on a given node, the resource's state will be `OFFLINE` on that node. If the `STOP` method exits nonzero or times out, then the state of the resource is `Stop_failed`.

Possible resource states include: `Online`, `Offline`, `Start_failed`, `Stop_failed`, `Monitor_failed`, `Online_not_monitored`, `Starting`, and `Stopping`.

Possible resource group states are: `Unmanaged`, `Online`, `Offline`, `Pending_online`, `Pending_offline`, `Error_stop_failed`, `Online_faulted`, and `Pending_online_blocked`.

In addition to resource state, the RGM also maintains a resource status that can be set by the resource itself by using the API. The field `Status Message` actually consists of two components: status keyword and status message. Status message is optionally set by the resource and is an arbitrary text string that is printed after the status keyword.

Descriptions of possible values for a resource's status are as follows:

`DEGRADED`	The resource is online, but its performance or availability might be compromised in some way.
`FAULTED`	The resource has encountered an error that prevents it from functioning.
`OFFLINE`	The resource is offline.
`ONLINE`	The resource is online and providing service.
`UNKNOWN`	The current status is unknown or is in transition.

Device Groups Device group status reflects the availability of the devices in that group.

The following are possible values for device group status and their descriptions:

`DEGRADED`	The device group is online, but not all of its potential primaries (secondaries) are up. For two-node connectivity, this status basically indicates that a stand-by primary does not exist, which means a failure of the primary node will result in a loss of access to the devices in the group.
`OFFLINE`	The device group is offline. There is no primary node. The device group must be brought online before any of its devices can be used.
`ONLINE`	The device group is online. There is a primary node, and devices within the group are ready for I/O.
`WAIT`	The device group is between one status and another. This status might occur, for example, when a device group is going from offline to online.

IP Network Multipathing Groups IP network multipathing (IPMP) group status reflects the availability of the backup group and the adapters in the group.

The following are possible values for IPMP group status and their descriptions:

`OFFLINE`	The backup group failed. All adapters in the group are offline.
`ONLINE`	The backup group is functional. At least one adapter in the group is online.

`UNKNOWN`	Any other state than those listed before. This could result when an adapter is detached or marked as down by Solaris commands such as `if_mpadm` or `ifconfig`.

The following are possible values for IPMP adapter status and their descriptions:

`OFFLINE`	The adapter failed or the backup group is offline.
`ONLINE`	The adapter is functional.
`STANDBY`	The adapter is on standby.
`UNKNOWN`	Any other state than those listed before. This could result when an adapter is detached or marked as down by Solaris commands such as `if_mpadm` or `ifconfig`.

Options You can specify command options to request the status for specific components.

If more than one option is specified, the `scstat` command prints the status in the specified order.

The following options are supported:

`-D` Shows status for all disk device groups.

You can use this option in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone. Output is the same when run from a zone as when run from the global zone.

You need `solaris.cluster.device.read` RBAC authorization to use this command option. See `rbac`(5).

`-g` Shows status for all resource groups.

You can use this option in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone. Output is the same when run from a zone as when run from the global zone.

You need `solaris.cluster.resource.read` RBAC authorization to use this command option. See `rbac`(5).

`-h` *node* Shows status for the specified node (*node*) and status of the disk device groups of which this node is the primary node. Also shows the status of the quorum devices to which this node holds reservations of the resource groups to which the node is a potential master, and holds reservations of the transport paths to which the *node* is attached.

You need `solaris.cluster.device.read`, `solaris.cluster.transport.read`, `solaris.cluster.resource.read`, `solaris.cluster.node.read`,

`solaris.cluster.quorum.read`, and `solaris.cluster.system.read` RBAC authorization to use this command option. See `rbac`(5).

`-i` Shows status for all IPMP groups and public network adapters.

You can use this option only in the global zone.

`-n` Shows status for all nodes.

You can use this option in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone. Output is the same when run from a zone as when run from the global zone.

You need `solaris.cluster.node.read` RBAC authorization to use this command option. See `rbac`(5).

`-p` Shows status for all components in the cluster. Use with `-v` to display more verbose output.

You can use this option in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone. Output is the same when run from a zone as when run from the global zone, except that no status for IPMP groups or public network adapters is displayed.

You need `solaris.cluster.device.read`, `solaris.cluster.transport.read`, `solaris.cluster.resource.read`, `solaris.cluster.node.read`, `solaris.cluster.quorum.read`, and `solaris.cluster.system.read` RBAC authorization to use `-p` with `-v`. See `rbac`(5).

`-q` Shows status for all device quorums and node quorums.

You can use this option in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone. Output is the same when run from a zone as when run from the global zone.

You need `solaris.cluster.quorum.read` RBAC authorization to use this command option. See `rbac`(5).

`-v[v]` Shows verbose output.

`-W` Shows status for cluster transport path.

You can use this option in the global zone or in a non-global zone. For ease of administration, use this form of the command in the global zone. Output is the same when run from a zone as when run from the global zone.

You need `solaris.cluster.transport.read` RBAC authorization to use this command option. See `rbac`(5).

Examples

EXAMPLE 1 Using the `scstat` Command

The following command displays the status of all resource groups followed by the status of all components related to the specified host:

```
% scstat -g -h host
```

The output that is displayed appears in the order in which the options are specified.

These results are the same results you would see by typing the two commands:

```
% scstat -g
```

and

```
% scstat -h host
```

Exit Status

The following exit values are returned:

`0`	The command completed successfully.
nonzero	An error has occurred.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Interface Stability	Evolving

See Also

Intro(1CL), cluster(1CL), `if_mpadm`(1M), `ifconfig`(1M), scha_resource_setstatus(1HA), scha_resource_setstatus(3HA), `attributes`(5)

Notes

An online quorum device means that the device was available for contributing to the formation of quorum when quorum was last established. From the context of the quorum algorithm, the device is online because it actively contributed to the formation of quorum. However, an online quorum device might not necessarily continue to be in a healthy enough state to contribute to the formation of quorum when quorum is re-established. The current version of Sun Cluster does not include a disk monitoring facility or regular probes to the quorum devices.

Name scswitch – perform ownership and state change of resource groups and device groups in Sun Cluster configurations

Synopsis

```
scswitch -c -h node[:zone][,...] -j resource[,...] -f flag-name

scswitch {-e | -n} [-M] -j resource[,...] [-h node[:zone][,...]]

scswitch -F {-g resource-grp[,...] | -D device-group[,...]}

scswitch -m -D device-group[,...]

scswitch -Q [ -g resource-grp[,...]] [-k]

scswitch -R -h node[:zone][,...] -g resource-grp[,...]

scswitch -r [-g resource-grp[,...]]

scswitch -S -h node[:zone][,...] [-K continue_evac]

scswitch -s [-g resource-grp[,...]] [-k]

scswitch {-u | -o} -g resource-grp[,...]

scswitch -Z [-g resource-grp[,...]]

scswitch -z -D device-group[,...] -h node[:zone][,...]

scswitch -z [-g resource-grp[,...]] [-h node[:zone][,...]]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scswitch` command moves resource groups or device groups, also called disk device groups, to new primary nodes. It also provides options for evacuating all resource groups and device groups from a node by moving ownership elsewhere, bringing resource groups or device groups offline and online, enabling or disabling resources, switching resource groups to or from an `Unmanaged` state, or clearing error flags on resources.

You can run the `scswitch` command from any node in a Sun Cluster configuration. If a device group is offline, you can use `scswitch` to bring the device group online onto any host in the node list. However, once the device group is online, a switchover to a spare node is not permitted. Only one invocation of `scswitch` at a time is permitted.

Do not attempt to kill an `scswitch` operation that is already underway.

You can use some forms of this command in a non-global zone, referred to simply as a zone. For more information about valid uses of this command in zones, see the descriptions of the individual options. For ease of administration, use this command in the global zone.

Options

Basic Options The following basic options are supported. Options that you can use with some of these basic options are described in "Additional Options."

`-c` Clears the `-f` *flag-name* error flag on the specified set of resources on the specified nodes or zones. For the current release of Sun Cluster software, the `-c` option is *only* implemented for the `Stop_failed` resource state. Clearing the `Stop_failed` resource state places the resource into the offline state on the specified nodes or zones.

If you use this option in a non-global zone, this option successfully operates only on resources that can be mastered by that zone. If you use this option in the global zone, this option can operate on any resource. For ease of administration, use this form of the command in the global zone.

If the `Stop` method fails on a resource and the `Failover_mode` property of the resource is set to `Hard`, the Resource Group Manager (RGM) halts or reboots the node or zone to force the resource (and all other resources mastered by that node or zone) offline.

If the `Stop` method fails on a resource and the `Failover_mode` property is set to a value other than `Hard`, the individual resource goes into the `Stop_failed` resource state, and the resource group is placed into the `Error_stop_failed` state. A resource group in the `Error_stop_failed` state on any node cannot be brought online on any node, nor can it be edited (you cannot add or delete resources or change resource group properties or resource properties). You must clear the `Stop_failed` resource state by performing the procedure that is documented in the *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*.

Caution – Make sure that both the resource and its monitor are stopped on the specified node or zone before you clear the `Stop_failed` resource state. Clearing the `Stop_failed` resource state without fully killing the resource and its monitor can lead to more than one instance of the resource executing on the cluster simultaneously. If you are using shared storage, this situation can cause data corruption. If necessary, as a last resort, execute a `kill`(1) command on the associated processes.

`-e` Enables the specified *resources*.

If you use this option in a non-global zone, this option successfully operates only on resources that can be mastered by that zone. If you use this option in the global zone, this option can operate on any resource. For ease of administration, use this form of the command in the global zone.

Once you have enabled a resource, it goes online or offline depending on whether its resource group is online or offline.

You can specify the `-h` option with the `-e` option to enable a resource on only a specified subset of nodes or zones. If you omit the `-h` option, the specified resources are enabled on all nodes or zones.

`-F` Takes offline the specified resource groups (`-g`) or device groups (`-D`) on all nodes.

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

When you specify the `-F` option with the `-D` option, you can run the `-F` option only from the global zone.

When the `-F` option takes a device group offline, the associated VxVM disk group or Solaris Volume Manager disk set is deported or released by the primary node. Before a device group can be taken offline, all access to its devices must be stopped, and all dependent file systems must be unmounted. You can start an offline device group by issuing an explicit `scswitch` call, by accessing a device within the group, or by mounting a file system that depends on the group.

`-m` Takes the specified device groups offline from the cluster for maintenance. The resulting state survives reboots.

You can use this option only in the global zone.

Before a device group can be placed in maintenance mode, all access to its devices must be stopped, and all dependent file systems must be unmounted. If a device group is currently being accessed, the action fails and the specified device groups are not taken offline from the cluster.

Device groups are brought back online by using the `-z` option. Only explicit calls to the `scswitch` command can bring a device group out of maintenance mode.

`-n` Disables the specified resources.

If you use this option in a non-global zone, this option successfully operates only on resources that can be mastered by that zone. If you use this option in the global zone, this option can operate on any resource. For ease of administration, use this form of the command in the global zone.

A disabled resource that is online on its current masters is immediately brought offline from its current masters. The disabled resource remains offline regardless of the state of its resource group.

You can specify the `-h` option with the `-e` option to disable a resource on only a specified subset of nodes or zones. If you omit the `-h` option, the specified resources are disabled on all nodes or zones.

`-o` Takes the specified unmanaged resource groups out of the unmanaged state.

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

Once a resource group is in the managed state, the RGM attempts to bring the resource group online.

`-Q` Brings the specified resource groups to a quiescent state.

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

If you omit the `-g` option, the `-Q` option applies to all resource groups.

This option stops the specified resource groups from continuously switching from one node to another in the event of the failure of a `Start` or `Stop` method. This form of the `scswitch` command does not exit until the resource groups have reached a quiescent state in which they are no longer stopping or starting on any node.

If a `Monitor_stop`, `Stop`, `Postnet_stop`, `Start`, or `Prenet_start` method fails on any resource in a group while the `scswitch -Q` command is executing, the resource behaves as if its `Failover_mode` property was set to `None`, regardless of its actual setting. Upon failure of one of these methods, the resource moves to an error state (either the `Start_failed` or `Stop_failed` resource state) rather than initiating a failover or a reboot of the node.

When the `scswitch -Q` command exits, the specified resource groups might be online or offline or in the `ONLINE_FAULTED` or `ERROR_STOPPED_FAILED` state. You can determine their current state by executing the `clresourcegroup status` command.

If a node dies during execution of the `scswitch -Q` command, execution might be interrupted, leaving the resource groups in a non-quiescent state. If execution is interrupted, `scswitch -Q` returns a nonzero exit code and writes an error message to the standard error. In this case, you can reissue the `scswitch -Q` command.

You can specify the `-k` option with the `-Q` option to hasten the quiescing of the resource groups. If you specify the `-k` option, it immediately kills all methods that are running on behalf of resources in the affected resource groups. If you do not specify the `-k` option, methods are allowed to continue running until they exit or exceed their configured timeout.

`-R` Takes the specified resource groups offline and then back online on the specified primary nodes or zones.

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

The specified node or zone must be a current primary node of the resource group.

`-r` Resumes the automatic recovery actions on the specified resource group, which were previously suspended by the `-s` option.

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

If you omit the `-g` option, the `-r` option applies to all resource groups.

A suspended resource group is *not* automatically restarted or failed over until you explicitly issue the command that resumes automatic recovery. Whether online or offline, suspended data services remain in their current state. You can still manually switch the resource group to a different state on specified nodes or zones. You can also still enable or disable individual resources in the resource group.

For information about how to suspend automatic recovery actions on resource groups, see the description of the `-s` option.

`-S` Switches all resource groups and device groups off the specified *node*, or switches all resource groups off the specified *zone*.

When used on a non-global zone, this option evacuates only the resource groups that are located in that zone. There is no effect on device groups.

When executed in a global zone, this option can evacuate any specified node or zone in the cluster. When executed in a non-global zone, this option can only evacuate that non-global zone.

The system attempts to select new primaries based on configured preferences for each group. All evacuated groups are not necessarily remastered by the same primary. If all groups that are mastered by the specified node or zone cannot be successfully evacuated from the specified node or zone, the command exits with an error.

Resource groups are first taken offline before they are relocated to new primary nodes or zones. An evacuated resource group might remain offline if the system cannot start it on a new primary node or zone.

If the primary ownership of a device group cannot be changed to one of the other nodes or zones, primary ownership for that device group is retained by the original node or zone.

`-s` Suspends the automatic recovery actions on and quiesces the specified resource group.

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

If you omit the `-g` option, the `-s` option applies to all resource groups.

A suspended resource group is not automatically started, restarted, or failed over until you explicitly resume monitoring of the resource group with this option. While monitoring of the resource group remains suspended, data services remain online. You can still manually switch the resource group online or offline on specified nodes or zones. You can also still enable or disable individual resources in the resource group.

You might need to suspend the automatic recovery of a resource group to investigate and fix a problem in the cluster. Or, you might need to perform maintenance on resource group services.

You can also specify the `-k` option to immediately kill all methods that are running on behalf of resources in the affected resource groups. By using the `-k` option, you can speed the quiescing of the resource groups. If you do not specify the `-k` option, methods are allowed to continue running until they exit or exceed their configured timeout.

For information about how to resume automatic recovery actions on resource groups, see the description of the `-r` option.

`-u` Puts the specified managed resource groups into the unmanaged state.

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

As a precondition of the `-u` option, all resources that belong to the indicated resource groups must first be disabled.

`-Z` This option does the following:

- Enables all resources of the specified resource groups
- Moves those resource groups into the managed state
- Brings those resource groups online on all the default primaries

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

If you omit the `-g` option, the `-Z` option applies to all resource groups.

When the `-g` option is not specified, the `scswitch` command attempts to bring all resource groups online, except resource groups that are suspended.

`-z` Requests a change in mastery of the specified resource group or device group.

If you use this option in a non-global zone, this option successfully operates only on resource groups whose node list contains that zone. If you use this option in the global zone, this option can operate on any resource group. For ease of administration, use this form of the command in the global zone.

If you omit the `-g` option, the `-z` option applies to all resource groups.

When used with the `-D` option, the `-z` option switches one or more specified device groups to the specified node. Only one primary node name can be specified for a device group's switchover. When multiple device groups are specified, the `-D` option switches the device groups in the order specified. If the `-z -D` operation encounters an error, the operation stops and no further switches are performed.

When used with only the `-g` option, the `-z` option brings the specified resource groups, which must already be managed, online on their most preferred nodes or zones. This form of `scswitch` does not bring a resource group online in violation of its strong `RG_affinities`, and it writes a warning message if the affinities of a resource group cannot be satisfied on any node or zone. This option does not enable any resources, enable monitoring on any resources, or take any resource groups out of the unmanaged state, as the `-Z` option does.

When used with the `-g` and `-h` options, the `-z` option brings the specified resource groups online on the nodes or zones that are specified by the `-h` option, and it takes them offline on all other cluster nodes or zones. If the node list that is specified with the `-h` option is empty (`-h ""`), the `-z` option takes the resource groups that are specified by the `-g` option offline from all of their current masters. All nodes or zones that are specified by the `-h` option must be current members of the cluster and must be potential primaries of all of the resource groups that are specified by the `-g` option. The number of nodes or zones that are specified by the `-h` option must not exceed the setting of the `Maximum_primaries` property of any of the resource groups that are specified by the `-g` option.

When used alone (`scswitch -z`), the `-z` option switches online all managed resource groups that are not suspended on their most preferred nodes or zones.

If you configure the `RG_affinities` property of one or more resource groups and you issue the `scswitch -z -g` command (with or without the `-h` option), additional resource groups other than those that are specified after the `-g` option might be switched as well. `RG_affinities` is described in rg_properties(5).

Additional Options

You can combine the following additional options with the previous basic options as follows:

`-D`

Specifies the name of one or more device groups.

This option is only legal with the `-F`, `-m`, and `-z` options.

You need `solaris.cluster.device.admin` role-based access control (RBAC) authorization to use this command option with the `-F`, `-m`, or `-z` option (in conjunction with the `-h` option). See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`, `pfcsh`, or `pfksh` profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su` to assume a role. You can also use `pfexec` to issue privileged Sun Cluster commands.

`-f`

Specifies the error *flag-name*.

This option is only legal with the `-c` option.

The only error flag that is currently supported is `Stop_failed`.

You need `solaris.cluster.resource.admin` RBAC authorization to use this command option with the `-c` option. See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`, `pfcsh`, or `pfksh` profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su` to assume a role. You can also use `pfexec` to issue privileged Sun Cluster commands.

`-g` Specifies the name of one or more resource groups.

This option is legal only with the `-F`, `-o`, `-Q`, `-r`, `-R`, `-s`, `-u`, `-z`, and `-Z` options.

You need `solaris.cluster.resource.admin` RBAC authorization to use this command option with the following options:

- `-F` option
- `-o` option
- `-Q` option
- `-R` option in conjunction with the `-h` option
- `-r` option
- `-s` option
- `-u` option
- `-Z` option
- `-z` option in conjunction with the `-h` option

See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`, `pfcsh`, or `pfksh` profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su` to assume a role. You can also use `pfexec` to issue privileged Sun Cluster commands.

`-h` Specifies the name of one or more cluster nodes or zones.

This option is only legal with the `-c`, `-e`, `-n`, `-R`, `-S`, and `-z` options.

When used with the `-c`, `-e`, `-n`, `-R`, or `-z` option, the `-h` option accepts a comma-delimited list of nodes or zones. The specified zones must be in the node list for the specified resource group or in the node list for the resource group that contains the specified resource.

To specify an empty node list to the `-z` option, specify two double quotation marks (`""`) as the argument to the `-h` option.

To specify a non-global zone, use the following syntax:

node:*zone*

The *node* component is the name of the physical node where *zone* is located. The *zone* component is the name of the zone that you want to include in `Nodelist`. For example, to specify the non-global zone `zone-1` which is located on the node `phys-schost-1`, you specify the following text:

```
phys-schost-1:zone-1
```

For resource groups that are configured with multiple primaries, the node or zone names that the `-h` option lists must all be valid potential primaries of each resource group that the `-g` option specifies.

If a resource group fails to start successfully on the node or zone that the `-h` option specifies, the resource group might fail over to a different node or zone. This behavior is determined by the setting of the `Failover_mode` resource property. See r_properties(5) for more information.

When used with the `-S` option, the `-h` option specifies the name of a single node from which to evacuate resource groups and device groups, or the name of a single zone from which to evacuate resource groups only.

You need `solaris.cluster.resource.admin` RBAC authorization to use this command option with the `-c`, `-R` option (in conjunction with the `-g` option), `-S`, or `-z` option (in conjunction with the `-g` option). In addition, you need `solaris.cluster.device.admin` RBAC authorization to use this command option with the `-z` option (in conjunction with the `-D` option). See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`, `pfcsh`, or `pfksh` profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su` to assume a role. You can also use `pfexec` to issue privileged Sun Cluster commands.

`-j` Specifies the names of one or more *resources*.

This option is legal only with the `-c`, `-e`, and `-n` options.

You need `solaris.cluster.resource.admin` RBAC authorization to use this command option with the `-c`, `-e`, or `-n` option. See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue

privileged Sun Cluster commands on the command line from the `pfsh`, `pfcsh`, or `pfksh` profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su` to assume a role. You can also use `pfexec` to issue privileged Sun Cluster commands.

`-K` Specifies the number of seconds to keep resource groups from switching back onto a node or zone after that node or zone has been successfully evacuated.

Resource groups cannot fail over or automatically switch over onto the node or zone while that node or zone is being evacuated, and, after evacuation is completed, for the number of seconds that you specify with this option. You can, however, initiate a switchover onto the evacuated node or zone with the `scswitch -z -g -h` command before *continue_evac* seconds have passed. Only automatic switchovers are prevented.

This option is legal only with the `-S` option. You must specify an integer value between 0 and 65535. If you do not specify a value, 60 seconds is used by default.

You need `solaris.cluster.resource.admin` RBAC authorization to use this command option. See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`, `pfcsh`, or `pfksh` profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su` to assume a role. You can also use `pfexec` to issue privileged Sun Cluster commands.

`-k` Immediately kills Resource Group Manager (RGM) resource methods that are running on behalf of resources in the specified resource groups.

You can use this option with the `-Q` and `-s` options. If you do not specify the `-k` option, methods are allowed to continue running until they exit or they exceed their configured timeout.

`-M` Enables (`-e`) or disables (`-n`) monitoring for the specified resources. When you disable a resource, you need not disable monitoring on it because both the resource and its monitor are kept offline.

This option is legal only with the `-e` and `-n` options.

You need `solaris.cluster.resource.admin` RBAC authorization to use this command option with the `-e` or `-n` option. See `rbac`(5).

You must also be able to assume a role to which the Sun Cluster Commands rights profile has been assigned to use this command. Authorized users can issue privileged Sun Cluster commands on the command line from the `pfsh`, `pfcsh`, or `pfksh` profile shell. A profile shell is a special kind of shell that enables you to access privileged Sun Cluster commands that are assigned to the Sun Cluster Commands rights profile. A profile shell is launched when you run `su` to assume a role. You can also use `pfexec` to issue privileged Sun Cluster commands.

Examples

EXAMPLE 1 Switching Over a Resource Group

The following command switches over `resource-grp-2` to be mastered by `schost-1`.

```
schost-1# scswitch –z –h schost-1 –g resource-grp-2
```

EXAMPLE 2 Bringing Online a Managed Resource Group Without Enabling Monitoring or Resources

The following command brings `resource-grp-2` online if `resource-grp-2` is already managed, but does not enable any resources or enable monitoring on any resources that are currently disabled.

```
schost-1# scswitch –z –g resource-grp-2
```

EXAMPLE 3 Switching Over a Resource Group Configured to Have Multiple Primaries

The following command switches over `resource-grp-3`, a resource group that is configured to have multiple primaries, to be mastered by `schost-1,schost-2,schost-3`.

```
schost-1# scswitch –z –h schost-1,schost-2,schost-3 –g resource-grp-3
```

EXAMPLE 4 Moving All Resource Groups and Device Groups Off a Node

The following command switches over all resource groups and device groups from `schost-1` to a new set of primaries.

```
schost-1# scswitch –S –h schost-1
```

EXAMPLE 5 Moving All Resource Groups and Device Groups Persistently Off a Node

The following command switches over all resource groups and device groups from `schost-1` to a new set of primaries. The command also specifies a 120-second wait before resource groups and device groups are permitted to switch back to `schost-1`.

The use of the `-K` option in the following command prevents resource groups from automatically switching back to `schost-1` after `schost-1` is successfully evacuated. An example of when a resource group might attempt to switch back to `schost-1` is if the resource

EXAMPLE 5 Moving All Resource Groups and Device Groups Persistently Off a Node *(Continued)*

group fails to start on its new master. Another example is if a resource group has strong negative affinities configured with the `RG_affinities` property.

```
schost-1# scswitch –S –h schost-1 -K 120
```

EXAMPLE 6 Restarting Resource Groups

The following command restarts `resource-grp-1` and `resource-grp-2` on the non-global zones `schost-1:zone1` and `schost-2:zone1`.

```
schost-1# scswitch –R –h schost-1:zone1,schost-2:zone1 \
–g resource-grp-1,resource-grp-2
```

EXAMPLE 7 Disabling Resources

```
schost-1# scswitch –n –j resource-1,resource-2
```

EXAMPLE 8 Enabling a Resource

```
schost-1# scswitch –e –j resource-1
```

EXAMPLE 9 Taking Resource Groups to the Unmanaged State

```
schost-1# scswitch –u –g resource-grp-1,resource-grp-2
```

EXAMPLE 10 Taking Resource Groups Out of the Unmanaged State

```
schost-1# scswitch –o –g resource-grp-1,resource-grp-2
```

EXAMPLE 11 Switching Over a Device Group

The following command switches over `device-group-1` to be mastered by `schost-2`.

```
schost-1# scswitch –z –h schost-2 –D device-group-1
```

EXAMPLE 12 Putting a Device Group Into Maintenance Mode

The following command puts `device-group-1` into maintenance mode.

```
schost-1# scswitch –m –D device-group-1
```

EXAMPLE 13 Quiescing Resource Groups

The following command brings resource groups `RG1` and `RG2` to a quiescent state.

```
schost-1# scswitch –Q -g RG1,RG2
```

EXAMPLE 14 Clearing a `Start_failed` Resource State by Switching Over a Resource Group

The `Start_failed` resource state indicates that a `Start` or `Prenet_start` method failed or timed out on a resource, but its resource group came online anyway. The resource group comes online even though the resource has been placed in a faulted state and might not be providing service. This state can occur if the resource's `Failover_mode` property is set to `None` or to another value that prevents the failover of the resource group.

Unlike the `Stop_failed` resource state, the `Start_failed` resource state does *not* prevent you or the Sun Cluster software from performing actions on the resource group. You do not need to issue the `scswitch -c` command to clear a `Start_failed` resource state. You only need to execute a command that restarts the resource.

The following command clears a `Start_failed` resource state that has occurred on a resource in the `resource-grp-2` resource group. The command clears this condition by switching the resource group to the `schost-2` node.

```
schost-1# scswitch –z –h schost-2 –g resource-grp-2
```

EXAMPLE 15 Clearing a `Start_failed` Resource State by Restarting a Resource Group

The following command clears a `Start_failed` resource state that has occurred on a resource in the `resource-grp-2` resource group. The command clears this condition by restarting the resource group on the `schost-1` node.

For more information about the `Start_failed` resource state, see the r_properties(5) man page.

```
schost-1# scswitch –R –h schost-1 –g resource-grp-2
```

EXAMPLE 16 Clearing a `Start_failed` Resource State by Disabling and Enabling a Resource

The following command clears a `Start_failed` resource state that has occurred on the resource `resource-1` by disabling and then re-enabling the resource.

For more information about the `Start_failed` resource state, see the r_properties(5) man page.

```
schost-1# scswitch –n –j resource-1
schost-1# scswitch –e –j resource-1
```

Exit Status This command blocks until requested actions are completely finished or an error occurs.

The following exit values are returned:

`0` The command completed successfully.

nonzero — An error has occurred. `scswitch` writes an error message to the standard error.

If the `scswitch` command exits with a nonzero exit status and the error message "`cluster is reconfiguring`" is displayed, the requested operation might have completed successfully, despite the error. If you doubt the result, you can execute the `scswitch` command again with the same arguments after the reconfiguration is complete.

If the `scswitch` command exits with a nonzero exit status and the error message "`Resource group failed to start on chosen node and may fail over to other node(s)`" is displayed, the resource group continues to reconfigure for some time after the `scswitch` command exits. Additional `scswitch` or `clresourcegroup` operations on that resource group fail until the resource group has reached a terminal state such as the `Online`, `Online_faulted`, or `Offline` state on all nodes.

If you invoke the `scswitch` command on multiple resources or resource groups and multiple errors occur, the exit value might only reflect one of the errors. To avoid this possibility, invoke the `scswitch` command on just one resource or resource group at a time.

Some operations are not permitted on a resource group (and its resources) whose `RG_system` property is `True`. See rg_properties(5) for more information.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWsczu`
Interface Stability	Evolving

See Also

`kill`(1), `pfcsh`(1), `pfexec`(1), `pfksh`(1), `pfsh`(1), Intro(1CL), cldevicegroup(1CL), clresourcegroup(1CL), `su`(1M), `attributes`(5), `rbac`(5), r_properties(5), rg_properties(5)

Sun Cluster Data Services Planning and Administration Guide for Solaris OS

Warnings

If you take a resource group offline by using the `-z` or `-F` option with the `-g` option, the `Offline` state of the resource group does not survive node reboots. If a node dies or joins the cluster, or if other resource groups are switching over, the resource group might come online. The resource group comes online on a node or zone even if you previously switched the resource group offline. Even if all of the resources are disabled, the resource group comes online.

To prevent the resource group from coming online automatically, use the `-s` option to suspend the automatic recovery actions of the resource group. To resume automatic recovery actions, use the `-r` option.

Name sctelemetry – initialize system resource monitoring

Synopsis

```
sctelemetry -d

sctelemetry -e

sctelemetry -i -o hasp_rg=rg,hasp_rs=rs [,hasp_mnt_pt=mnt_pt] [,db_rg=rg] [,db_rs=rs]
[,telemetry_rg=rg] [,telemetry_rs=rs]

sctelemetry -i -o hasp_mnt_pt=mnt_pt,hasp_nodelist=node[:...] [,hasp_rs=rs] [,db_rg=rg]
[,db_rs=rs] [,telemetry_rg=rg,telemetry_rs=rs]

sctelemetry -u
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `sctelemetry` command initializes system resource monitoring, brings monitoring online, and takes it offline. When initializing, use the `-o` option with the `hasp_rg=`*rg*,`hasp_rs=`*rs* parameters to rely on an existing resource of type `SUNW.HAStoragePlus`. Use the `-o` option with the `hasp_mnt_pt=`*mnt_pt*,`hasp_nodelist=`*node*`[:,...]`parameters to have the `sctelemetry` command create a resource of type `SUNW.HAStoragePlus`. For more information about the resource types, see the SUNW.derby(5), SUNW.HAStoragePlus(5), and SUNW.SCTelemetry(5) man pages.

`SUNW.SCTelemetry` is instantiated in a multi-master resource group, that is the resource group is configured on all cluster nodes and does not use network load balancing.

You can use this command only in the global zone.

Options The options for `sctelemetry` are as follows:

`-d`
Disables the collection of system resource usage data and the database in which telemetry data is stored.

You can use this option only in the global zone.

Users other than superuser require `solaris.cluster.system.modify` RBAC authorization to use the `-d` option of `sctelemetry`. For more information, see the `rbac`(5) man page.

`-e`
Brings collection of system resource usage data online. By default, system resource monitoring is online when you use the `-i` option of the `sctelemetry` command.

You can use this option only in the global zone.

Users other than superuser require `solaris.cluster.system.modify` RBAC authorization to use the `-e` option of `sctelemetry`. For more information, see the `rbac`(5) man page.

`-i`

Creates resource groups containing resources of type `SUNW.SCTelemetry` and `SUNW.derby`. By default, when you create these resources and resource groups by using the `-i` option, system resource monitoring is online.

You can use this option only in the global zone.

Users other than superuser require `solaris.cluster.system.modify` RBAC authorization to use the `-i` option of `sctelemetry`. For more information, see the `rbac`(5) man page.

`-o` hasp_rg=*rg*,hasp_rs=*rs*[,hasp_mnt_pt=*mnt_pt*][,db_rg=*rg*][,db_rs=*rs*][,telemetry_rg=*rg*][,telemetry_rs=*rs*]

When used with the `-i` option, identifies the resource of type `SUNW.HAStoragePlus` to be used by the database and the resource group that contains this resource. The data collection facility must have access to a file system for `SUNW.HAStoragePlus`.

The parameters are as follows:

hasp_rg=*rg*	The resource group that contains the resource of type `SUNW.HAStoragePlus` that is used for system resource monitoring. You must specify *rg*, the name of this resource group.
hasp_rs=*rs*	The resource of type `SUNW.HAStoragePlus` that is used for system resource monitoring. You must specify *rs*, the name of this resource.
hasp_mnt_pt=*mnt_pt*	The mount point on which `sctelemetry` stores database files for system resource monitoring. This mount point must be a property of the resource, `hasp_rs`. Specifying this mount point is obligatory if there is more than one mount point in `hasp_rs`.
db_rg=*rg*	The resource group in which `sctelemetry` configures the resource of type `SUNW.derby`. You can specify *rg*, the name of this resource group.
db_rs=*rs*	The resource of type `SUNW.derby` that `sctelemetry` configures. You can specify *rs*, the name of this resource.
telemetry_rg=*rg*	The resource group in which `sctelemetry` configures a resource of type `SUNW.SCTelemetry`. You can specify *rg*, the name of this resource group.

telemetry_rs=*rs* — The resource of type `SUNW.SCTelemetry` that `sctelemetry` configures. You can specify *rs*, the name of this resource.

`-o` hasp_mnt_pt=*mnt_pt*,hasp_nodelist=*node*[:...][,hasp_rs=*rs*][,db_rg=*rg*][,db_rs=*rs*]\
[,telemetry_rg=*rg*][,telemetry_rs=*rs*]

When used with the `-i` option, specifies the nodes on which the `SUNW.HAStoragePlus` file system for data collection is accessible and specifies the mount point for the file system in which Sun Cluster stores system resource data.

The parameters are as follows:

hasp_mnt_pt=*mnt_pt* — The mount point that `sctelemetry` uses to configure a resource of type `SUNW.HAStoragePlus`. You must specify *mnt_pt*, the name of the mount point. The shared storage must be configured before the `HAStoragePlus` resource to be created. This mount point refers to the shared storage and must appear in `/etc/vfstab` as follows:

```
/dev/md/ddg/dsk/d20 /dev/md/ddg/rdsk/d20    \
/mntpt ufs     2       no logging
```

hasp_nodelist=*node*[:...] — The nodes with which `sctelemetry` configures a resource of type `SUNW.HAStoragePlus`. You must specify *node*:[...], the name of the nodes.

hasp_rs=*rs* — The resource of type `SUNW.HAStoragePlus` that `sctelemetry` configures. You can specify *rs*, the name of this resource.

db_rg=*rg* — The resource group in which `sctelemetry` configures a resource of type `SUNW.derby`. You can specify *rg*, the name of this resource group.

db_rs=*rs* — The resource of type `SUNW.derby` that `sctelemetry` configures. You can specify *rs*, the name of this resource.

telemetry_rg=*rg* — The resource group in which `sctelemetry` configures a resource of type `SUNW.SCTelemetry`. You can specify *rg*, the name of this resource group.

telemetry_rs=*rs* — The resource of type `SUNW.SCTelemetry` that `sctelemetry` configures. You can specify *rs*, the name of this resource.

`-u`

Removes the resources and resource groups that were previously created by using the `-i` option.

You can use this option only in the global zone.

Users other than superuser require `solaris.cluster.system.modify` RBAC authorization to use the `-u` option of the `sctelemetry` command. For more information, see the `rbac`(5) man page.

Examples

Initializing System-Resource Monitoring, When a `HAStoragePlus` Resource Exists

This example initializes system-resource monitoring and verifies that monitoring has been initialized. This example assumes that you have a `SUNW.HAStoragePlus` resource available for system-resource monitoring.

This example does not specify the names of the resources *db_rs* and *telemetry_rs* or the resource groups *db_rg* and *telemetry_rg*. The `sctelemetry` command gives these resources and resource groups default names.

The output of the `scstat -g` command shows the relationship between resources and resource groups involved in system resource monitoring. The output also shows that the *db_rs* and *hasp_rs* resources and the *db_rg* resource group are each `online` on one node, the *telemetry_rg* and *telemetry_rs* are `online` on all cluster nodes.

```
# sctelemetry -i \
-o hasp_mnt_pt=DBDATA,hasp_nodelist=l6-lx-1:l6-lx-4,hasp_rs=anto

# scstat -g
```

Disabling System-Resource Monitoring

This example disables system-resource monitoring then verifies that the monitoring has been disabled. When monitoring is disabled, the output of the `scstat -g` command shows that the *db_rs*, *hasp_rs*, and *telemetry_rs* resources and the *db_rg* and *telemetry_rg* resource groups are `offline`.

```
# sctelemetry -d
# scstat -g
```

Exit Status

The following exit values are returned:

`0`	The command completed successfully.
nonzero	An error has occurred.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

Intro(1CL), cltelemetryattribute(1CL), cluster(1CL), scstat(1M), sctelemetry(1M), SUNW.derby(5), SUNW.HAStoragePlus(5), SUNW.SCTelemetry(5)

Name scversions – Sun Cluster version management

Synopsis

```
scversions [-c]
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scversions` command commits the cluster to a new level of functionality after a rolling–upgrade to new Sun Cluster software. With no arguments, the `scversions` command prints a message indicating whether a commitment is needed.

Operands The following operands are supported:

`-c` Commit the set of nodes that are currently active members of the cluster to the highest possible level of functionality.

When you upgrade a node (either through upgrade to a new release of the product or by application of a patch) and boot it back into the cluster, some of the internal protocols on that node might have to run at lower versions in order to cooperate correctly with other nodes in the cluster. When the cluster is in this state, some administrative actions might be disabled and some new functionality introduced in the upgrade might be unavailable.

When you run this command once from any node after all nodes are upgraded, the cluster switches to the highest versions of internal protocols possible. Assuming that all nodes have the same Sun Cluster software installed at that time, all new functionality becomes available and any administrative restrictions are removed.

If a node that has not been upgraded is an active member of the cluster at the time you run the `-c` option to `scversions`, the command has no effect because the cluster is already running at the highest possible level of functionality.

If a node has not been upgraded and is not an active member of the cluster when you run the `-c` option to `scversions` (for example, if that node is down for maintenance), the internal protocols of the cluster are upgraded to the highest possible versions. You might have to upgrade the node that was not an active member of the cluster to enable it to rejoin the cluster.

Exit Status

0	Success
non-zero	Failure

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also scinstall(1M)

Name scvxinstall – install VERITAS Volume Manager (VxVM) on a cluster node

Synopsis

```
scvxinstall [-d media-image] [-L license...]

scvxinstall -H

scvxinstall {-i | -e} [-d media-image] [-L license...]

scvxinstall -s
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `scvxinstall` utility provides automatic VxVM installation and optional root-disk encapsulation for Sun Cluster nodes.

The first form of the `scvxinstall` utility in the `SYNOPSIS` section of this man page runs in interactive mode. All other forms of the utility run in non-interactive mode.

- In interactive mode, `scvxinstall` prompts the user for the mode of operation ("install only" or "install and encapsulate") and for any needed CD-ROM and licensing information.
- In non-interactive mode, `scvxinstall` does not prompt the user for information. If any needed information is not supplied on the utility line, `scvxinstall` terminates with an error return code.

The cluster must meet the following requirements before you run the `scvxinstall` utility:

- All nodes in the cluster configuration must be current cluster members.
- Each root disk that you will encapsulate must have at least two free (unassigned) partitions.
- All nodes must be added to the node authentication list.
- As of VxVM 4.1, you must install VxVM software and licenses before you run the `scvxinstall` utility.

The "install-only" mode of the `scvxinstall` utility performs the following tasks:

1. Verifies that the node you are installing is booted in cluster mode and is running as root, and verify that all other cluster nodes are running in cluster mode.
2. For VxVM 4.0 or earlier, adds the VxVM software, licensing, and man-page packages, but not the GUI packages.
3. Negotiates a cluster-wide value for the `vxio` major number by modifying the `/etc/name_to_major` file. This ensures that the `vxio` number is the same on all cluster nodes.
4. For VxVM 4.0 or earlier, installs the VxVM license key.

5. Instructs the user to reboot the node to resume operation with the new `vxio` major numbers in effect.

The “install-and-encapsulate” mode of the `scvxinstall` utility performs the same tasks as the “install-only” mode **except** Step 5, then performs the following additional tasks:

1. Runs several VxVM commands to prepare for root-disk encapsulation.
2. Modifies the global-devices entry in the `/etc/vfstab` file specified for the `/global/.devices/node@`*n* file system, where *n* is the node ID number. The `scvxinstall` utility replaces the existing device path `/dev/did/{r}dsk` with `/dev/{r}dsk`. This change ensures that VxVM recognizes that the global-devices file system resides on the root disk.
3. Twice reboots each node that is running `scvxinstall`, once to allow VxVM to complete the encapsulation process and once more to resume normal operation. The `scvxinstall` utility includes a synchronization mechanism to ensure that it reboots only one node at a time, to prevent loss of quorum.
4. Unmounts the global-devices file system. The file system is automatically remounted after the encapsulation process is complete.
5. Recreates the special files for the root-disk volumes with a unique minor number on each node.

You can use this command only in the global zone.

Options The following options are supported:

`-d` *media-image*
Valid only for VxVM 4.0 or earlier. Specifies the path to the VxVM packages.

You can use this option only in the global zone.

`-e`
Specifies the "install and encapsulate" mode of the `scvxinstall` utility.

You can use this option only in the global zone.

This option installs VxVM, if installing VxVM 4.0 or earlier, encapsulates the root disk, and performs postinstallation tasks. If the `scvxinstall` utility was previously run on the node in "install only" mode, `scvxinstall` confirms that "install only" mode tasks are completed before it performs the root-disk encapsulation tasks.

`-H`
Specifies the "help" mode of the `scvxinstall` utility. This option displays a brief help message about the `scvxinstall` utility.

You can use this option only in the global zone.

`-i`
Specifies the “install only” mode of the `scvxinstall` utility.

You can use this option only in the global zone.

This option installs VxVM, if installing VxVM 4.0 or earlier, and performs postinstallation tasks, but does not encapsulate the root disk.

`-L` *license*
Valid only for VxVM 4.0 or earlier. Specifies a license key for the VxVM software. You can specify the `-L` *license* option multiple times to supply multiple license keys to the `scvxinstall` utility. If you have no additional license keys to install, you can specify the word `none` for the *license* argument to the `-L` option.

`-s`
Specifies the "show status" mode of the `scvxinstall` utility. This option displays the status of running or completed `scvxinstall` processing on the node.

You can use this option only in the global zone.

Examples

EXAMPLE 1 Running `scvxinstall` Interactively

The following command runs `scvxinstall` interactively.

```
example# scvxinstall
```

EXAMPLE 2 Installing the VxVM Packages Without Encapsulating the Root Disk

The following command installs the VxVM 4.0 packages but does not encapsulate the root disk. This command also supplies the VxVM license key. This example assumes that the VxVM CD-ROM is in the CD-ROM drive.

```
example# scvxinstall -i -L "9999 9999 9999 9999 9999 999"
```

EXAMPLE 3 Installing the VxVM Packages Without Encapsulating the Root Disk

The following command installs the VxVM 4.0 packages but does not encapsulate the root disk. The command supplies the path to the CD-ROM images of the VxVM packages, which are stored on a server.

```
example# scvxinstall -i -d /net/myserver/VxVM/pkgs
```

EXAMPLE 4 Installing the VxVM Packages and Encapsulating the Root Disk

The following command installs the VxVM 4.0 packages and encapsulates the root disk. The command supplies the VxVM license key. This example assumes that the VxVM CD-ROM is in the CD-ROM drive.

```
example# scvxinstall -e -L "9999 9999 9999 9999 9999 999"
```

EXAMPLE 5 Installing the VxVM Packages and Encapsulating the Root Disk

The following command installs the VxVM 4.0 packages and encapsulates the root disk. The command supplies the path to the CD-ROM images and supplies the VxVM license key.

```
example# scvxinstall -e -d /net/myserver/VxVM/pkgs -L "9999 9999 9999 9999 9999 999"
```

EXAMPLE 6 Encapsulating the Root Disk After Installing VxVM 4.1 Software

The following command verifies that VxVM 4.1 software and licenses are installed and configured, encapsulates the root disk, and negotiates a cluster-wide value for the `vxio` major number.

```
example# scvxinstall -e
```

EXAMPLE 7 Performing Postinstallation Tasks After Installing VxVM 4.1 Software

The following command verifies that VxVM 4.1 software and licenses are installed and configured, then negotiates a cluster-wide value for the `vxio` major number.

```
example# scvxinstall -i
```

Exit Status

The following exit values are returned:

`0`	Successful completion.
non-zero	An error has occurred.

Files

`/etc/rc2.d/S74scvxinstall.sh`
(Solaris 9) An `rc` script used to complete processing following a root-disk-encapsulation reboot

`/usr/cluster/lib/svc/method/scvxinstall`
(Solaris 10) An `rc` script used to complete processing following a root-disk-encapsulation reboot

`/var/cluster/logs/install/scvxinstall.log.`*pid*
Log file created by `scvxinstall`

`/var/cluster/scvxinstall/*`
Location of temporary files used by `scvxinstall`

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu, SUNWscr

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also Intro(1CL), cldevice(1CL), cldevicegroup(1CL), clsetup(1CL), cluster(1CL), clvxvm(1CL), scinstall(1M)

Sun Cluster Software Installation Guide for Solaris OS

Name sc_zonesd – Sun Cluster zone administration daemon

Synopsis

```
/usr/cluster/lib/sc/sc_zonesd
```

Description The `sc_zonesd` daemon is a system daemon that is used by Sun Cluster software to track and manage non-global zones. This daemon initially starts when the system comes up.

The daemon runs in the global zone only.

Diagnostics All diagnostic messages are logged through the `syslog` function.

Notes The `sc_zonesd` daemon must be started in superuser mode.

The `sc_zonesd` daemon is controlled by the SMF service `sc_zones`. The cluster software requires this daemon to run for non-global zone support. If the daemon is killed or if the SMF service is disabled, the cluster node will panic.

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu
Interface Stability	Private

See Also `syslog`(3C), `attributes`(5)

REFERENCE

SC32 3ha

Name scds_calls – Sun Cluster Data Services Development Library (DSDL) functions

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>
```

Description The Data Services Development Library (DSDL) is a set of higher-level library functions that encapsulate and extend the functionality of the `scha` library functions. The `scha` library functions are described in the scha_calls(3HA) man page.

DSDL functions are implemented in the `libdsdev.so` library.

DSDL functions are generally divided into the following categories.

- General-purpose functions General-purpose functions include initialization functions, retrieval functions, failover and restart functions, and execution functions. These functions enable you to perform the following operations:
  - Initialize the DSDL environment
  - Retrieve resource type, resource, and resource group names, and extension property values
  - Fail over and restart a resource group and restart a resource
  - Convert error strings to error messages
  - Execute a command under a timeout
- Property functions

  These functions provide convenience APIs for accessing specific properties of the relevant resource type, resource, and resource group, including some commonly used extension properties. The DSDL provides the `scds_initialize()` function to parse the command-line arguments. The library *caches* the various properties of the relevant resource type, resource, and resource group.
- Network resource access functions

  These functions manage network resources that are used by resources and resource groups. These functions handle host names, port lists, and network addresses, and they enable TCP-based monitoring.
- Process Monitor Facility (PMF) functions

  These functions encapsulate the Process Monitor Facility (PMF) functionality.
- Fault monitor functions

  These functions provide a predetermined model of fault monitoring by keeping the failure history and evaluating it in conjunction with the `Retry_count` and `Retry_interval` properties.
- Utility functions

  These functions enable you to write messages and debugging messages to the system log.

Initialization Functions

The following functions initialize the calling method:

- scds_initialize(3HA) – Allocates resources and initializes the DSDL environment.
- scds_close(3HA) – Frees resources that are allocated by the `scds_initialize()` function.

Retrieval Functions

The following functions retrieve information about zones, resource types, resources, resource groups, and extension properties:

- scds_get_zone_name(3HA) – Retrieves the name of the zone on whose behalf a method is running.
- scds_get_resource_type_name(3HA) – Retrieves the name of the resource type for the calling program.
- scds_get_resource_name(3HA) – Retrieves the name of the resource for the calling program.
- scds_get_resource_group_name(3HA) – Retrieves the name of the resource group for the calling program.
- scds_get_ext_property(3HA) – Retrieves the value of the specified extension property.
- scds_free_ext_property(3HA) – Frees the memory that is allocated by `scds_get_ext_property()`.

The following function retrieves status information about the `SUNW.HAStoragePlus` resources that are used by a resource:

scds_hasp_check(3HA) – Retrieves status information about `SUNW.HAStoragePlus` resources that are used by a resource. This information is obtained from the state (online or otherwise) of all `SUNW.HAStoragePlus` resources on which the resource depends by using the `Resource_dependencies` or `Resource_dependencies_weak` system properties that are defined for the resource. See the SUNW.HAStoragePlus(5) man page for more information.

Failover and Restart Functions

The following functions fail over or restart a resource or resource group:

- scds_failover_rg(3HA) – Fails over a resource group.
- scds_restart_rg(3HA) – Restarts a resource group.
- scds_restart_resource(3HA) – Restarts a resource.

Execution Functions

The following functions execute a command under a timeout and convert an error code to an error message:

- scds_timerun(3HA) – Executes a command under a timeout value.
- scds_error_string(3HA) and scds_error_string_i18n(3HA) – Translates an error code to an error string. Strings that are returned by `scds_error_string()` are displayed in English. Strings that are returned by `scds_error_string_i18n()` are displayed in the native language that is specified by the `LC_MESSAGES` locale category.
- scds_svc_wait(3HA) - Waits for the specified timeout period for a monitored process to die.

Property Functions

These functions provide convenience APIs for accessing specific properties of the relevant resource type, resource, and resource group, including some commonly used extension properties. The DSDL provides the `scds_initialize()` function to parse the command-line arguments. The library *caches* the various properties of the relevant resource type, resource, and resource group.

The scds_property_functions(3HA) man page describes these functions, which include the following:

- `scds_get_ext_`*property-name*
- `scds_get_rg_`*property-name*
- `scds_get_rs_`*property-name*
- `scds_get_rt_`*property-name*

Network Resource Access Functions

You use these functions to manage network resources.

The following functions handle host names:

- scds_get_rs_hostnames(3HA) – Retrieves a list of host names that is used by the resource.
- scds_get_rg_hostnames(3HA) – Retrieves a list of host names that is used by the network resources in a resource group.
- scds_print_net_list(3HA) – Writes the contents of the host name list to `syslog`(3C). You typically use this function for debugging.
- scds_free_net_list(3HA) – Frees the memory that is allocated by `scds_get_rs_hostnames()` or `scds_get_rg_hostnames()`.

The following functions handle port lists:

- scds_get_port_list(3HA) – Retrieves a list of port-protocol pairs that is used by a resource.
- scds_print_port_list(3HA) – Writes the contents of the port-protocol list to `syslog`(3C). You typically use this function for debugging.
- scds_free_port_list(3HA) – Frees the memory that is allocated by `scds_get_port_list()`.

The following functions handle network addresses:

- scds_get_netaddr_list(3HA) – Retrieves a list of network addresses that is used by a resource.
- scds_print_netaddr_list(3HA) – Writes the contents of the network address list to `syslog`(3C). You typically use this function for debugging.
- scds_free_netaddr_list(3HA) – Frees the memory that is allocated by `scds_get_netaddr_list()`.

The following functions enable TCP-based monitoring. Typically, a fault monitor uses these functions to establish a simple socket connection to a service, read and write data to the service to ascertain its status, and disconnect from the service.

This set of functions includes the following functions:

- scds_fm_tcp_connect(3HA) – Establishes a TCP connection to a process that uses IPv4 addressing only.
- scds_fm_net_connect(3HA) – Establishes a TCP connection to a process that uses either IPv4 or IPv6 addressing.
- scds_fm_tcp_read(3HA) – Uses a TCP connection to read data from the process that is being monitored.
- scds_fm_tcp_write(3HA) – Uses a TCP connection to write data to a process that is being monitored.
- scds_simple_probe(3HA) – Probes a process by establishing and terminating a TCP connection to the process. This function handles only IPv4 addresses.
- scds_simple_net_probe(3HA) – Probes a process by establishing and terminating a TCP connection to the process. This function handles either IPv4 or IPv6 addresses.
- scds_fm_tcp_disconnect(3HA) – Terminates the connection to a process that is being monitored. This function handles only IPv4 addresses.
- scds_fm_net_disconnect(3HA) – Terminates the connection to a process that is being monitored. This function handles either IPv4 or IPv6 addresses.

PMF Functions

These functions encapsulate the Process Monitor Facility (PMF) functionality. The DSDL model for monitoring through the PMF creates and uses implicit *tag* values for `pmfadm`. See the pmfadm(1M) man page for more information.

The PMF facility also uses implicit values for the `Restart_interval`, `Retry_count`, and `action_script` (the `-t`, `-n`, and `-a` options to `pmfadm`). Most important, the DSDL ties the process failure history, as determined by the PMF, into the application failure history as detected by the fault monitor to compute the restart or failover decision.

The set includes the following functions:

- scds_pmf_get_status(3HA) – Determines if the specified instance is being monitored under the PMF's control.
- scds_pmf_restart_fm(3HA) – Uses the PMF to restart the fault monitor.
- scds_pmf_signal(3HA) – Sends the specified signal to a process tree that is running under the PMF's control.
- scds_pmf_start(3HA) and scds_pmf_start_env(3HA) – Executes a specified program (including a fault monitor) under the PMF's control. In addition to performing the same operations as the `scds_pmf_start()` function, the `scds_pmf_start_env()` function also passes a provided environment to the executed program.
- scds_pmf_stop(3HA) – Terminates a process that is running under the PMF's control.
- scds_pmf_stop_monitoring(3HA) – Stops monitoring a process that is running under the PMF's control.

Fault Monitor Functions

These functions provide a predetermined model of fault monitoring by keeping the failure history and evaluating it in conjunction with the `Retry_count` and `Retry_interval` properties.

This set includes the following functions:

- scds_fm_sleep(3HA) – Waits for a message on a fault monitor control socket.
- scds_fm_action(3HA) – Takes action after a probe completes.
- scds_fm_print_probes(3HA) – Writes probe status information to the system log.

Utility Functions

The following functions enable you to write messages and debugging messages to the system log:

- scds_syslog(3HA) – Writes messages to the system log.
- scds_syslog_debug(3HA) – Writes debugging messages to the system log.

Files

`/usr/cluster/include/scds.h`	Include file
`/usr/cluster/lib/libdsdev.so`	Library

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Evolving

See Also

pmfadm(1M), scds_close(3HA), scds_error_string(3HA), scds_error_string_i18n(3HA), scds_failover_rg(3HA), scds_fm_action(3HA), scds_fm_net_connect(3HA), scds_fm_net_disconnect(3HA), scds_fm_print_probes(3HA), scds_fm_sleep(3HA), scds_fm_tcp_connect(3HA), scds_fm_tcp_disconnect(3HA), scds_fm_tcp_read(3HA), scds_fm_tcp_write(3HA), scds_free_ext_property(3HA), scds_free_net_list(3HA), scds_free_netaddr_list(3HA), scds_free_port_list(3HA), scds_get_ext_property(3HA), scds_get_netaddr_list(3HA), scds_get_port_list(3HA), scds_get_resource_group_name(3HA), scds_get_resource_name(3HA), scds_get_resource_type_name(3HA), scds_get_rg_hostnames(3HA), scds_get_rs_hostnames(3HA), scds_get_zone_name(3HA), scds_hasp_check(3HA), scds_initialize(3HA), scds_pmf_get_status(3HA), scds_pmf_restart_fm(3HA), scds_pmf_signal(3HA), scds_pmf_start(3HA), scds_pmf_stop(3HA), scds_pmf_stop_monitoring(3HA), scds_print_net_list(3HA), scds_print_netaddr_list(3HA), scds_print_port_list(3HA), scds_property_functions(3HA), scds_restart_resource(3HA), scds_restart_rg(3HA), scds_simple_net_probe(3HA), scds_simple_probe(3HA), scds_svc_wait(3HA), scds_syslog(3HA), scds_syslog_debug(3HA), bscds_timerun(3HA), scha_calls(3HA), SUNW.HAStoragePlus(5), `attributes`(5)

Name scds_close – free DSDL environment resources

Synopsis

```
cc [flags...]-I
/usr/cluster/include file -L /usr/cluster/lib  -l
dsdev
#include <rgm/libdsdev.h>

void scds_close(scds_handle_t*handle
```

Description The `scds_close()` function reclaims resources that were allocated during data service method initialization by using scds_initialize(3HA). Call this function once, prior to termination of the program.

Parameters The following parameters are supported:

handle The handle returned from `scds_initialize()`.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_initialize(3HA), `attributes`(5)

Name scds_error_string, scds_error_string_i18n – generate an error string from an error code

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

const char *scds_error_string(scha_err_t error_code

const char *scds_error_string_i18n(scha_err_t error_code
```

Description The `scds_error_string()` and `scds_error_string_i18n()` functions generate a short string that describes an error from an error code that is returned by a DSDL function. Strings that are returned by `scds_error_string()` are displayed in English. Strings that are returned by `scds_error_string_i18n()` are displayed in the native language that is specified by the `LC_MESSAGES` locale category. See `setlocale`(3C). Invalid error codes return `NULL`.

The pointer that is returned by this function points to memory that belongs to the DSDL. Do not modify this memory.

Parameters The following parameters are supported:

error_code Error code that is returned by a DSDL function.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scha_calls(3HA), `setlocale`(3C), `attributes`(5)

Name scds_failover_rg – failover a resource group

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_failover_rg(scds_handle_t handle);
```

Description The `scds_failover_rg()` function performs a scha_control(3HA) `SCHA_GIVEOVER` operation on the resource group containing the resource passed to the calling program.

When this function succeeds, it does not return. Therefore, treat this function as the last piece of code to be executed in the calling program.

Parameters The following parameters are supported:

handle The handle that is returned from scds_initialize(3HA).

Return Values The following return values are supported:

`SCHA_ERR_NOERR` Indicates the function succeeded.

Other values Indicate the function failed. See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scha_calls(3HA), scha_control(3HA), `attributes`(5)

Name scds_fm_action – take action after probe completion function

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_fm_action(scds_handle_t handle, int probe_status, long elapsed_milliseconds
```

Description The `scds_fm_action()` function uses the `probe_status` of the data service in conjunction with the past history of failures to take one of the following actions:

- Restart the application.
- Fail over the resource group.
- Do nothing.

Use the value of the input `probe_status` argument to indicate the severity of the failure. For example, you might consider a failure to connect to an application as a complete failure, but a failure to disconnect as a partial failure. In the latter case you would have to specify a value for `probe_status` between 0 and `SCDS_PROBE_COMPLETE_FAILURE`.

The DSDL defines `SCDS_PROBE_COMPLETE_FAILURE` as `100`. For partial probe success or failure, use a value between `0` and `SCDS_PROBE_COMPLETE_FAILURE`.

Successive calls to `scds_fm_action()` compute a failure history by summing the value of the `probe_status` input parameter over the time interval defined by the `Retry_interval` property of the resource. Any failure history older than `Retry_interval` is purged from memory and is not used towards making the restart or failover decision.

The `scds_fm_action()` function uses the following algorithm to choose which action to take:

Restart If the accumulated history of failures reaches `SCDS_PROBE_COMPLETE_FAILURE`, `scds_fm_action()` restarts the resource by calling the `STOP` method of the resource followed by the `START` method. It ignores any `PRENET_START` or `POSTNET_STOP` methods defined for the resource type.

The status of the resource is set to `SCHA_RSSTATUS_DEGRADED` by making a `scha_resource_setstatus()` call, unless the resource is already set.

If the restart attempt fails because the `START` or `STOP` methods of the resource fail, a `scha_control()` is called with the `GIVEOVER` option to fail the resource group over to another node or zone. If the `scha_control()` call succeeds, the resource group is failed over to another cluster node or zone, and the call to `scds_fm_action()` never returns.

Upon a successful restart, failure history is purged. Another restart is attempted only if the failure history again accumulates to `SCDS_PROBE_COMPLETE_FAILURE`.

Failover If the number of restarts attempted by successive calls to `scds_fm_action()` reaches the `Retry_count` value defined for the resource, a failover is attempted by making a call to `scha_control()` with the `GIVEOVER` option.

The status of the resource is set to `SCHA_RSSTATUS_FAULTED` by making a `scha_resource_setstatus()` call, unless the resource is already set.

If the `scha_control()` call fails, the entire failure history maintained by `scds_fm_action()` is purged.

If the `scha_control()` call succeeds, the resource group is failed over to another cluster node or zone, and the call to `scds_fm_action()` never returns.

No Action If the accumulated history of failures remains below `SCDS_PROBE_COMPLETE_FAILURE`, no action is taken. In addition, if the `probe_status` value is `0`, which indicates a successful check of the service, no action is taken, irrespective of the failure history.

The status of the resource is set to `SCHA_RSSTATUS_OK` by making a `scha_resource_setstatus()` call, unless the resource is already set.

Parameters The following parameters are supported:

handle The handle that is returned from scds_initialize(3HA).

probe_status A number you specify between 0 and `SCDS_PROBE_COMPLETE_FAILURE` that indicates the status of the data service. A value of `0` implies that the recent data service check was successful. A value of `SCDS_PROBE_COMPLETE_FAILURE` means complete failure and implies that the service has completely failed. You can also supply a value in between 0 and `SCDS_PROBE_COMPLETE_FAILURE` that implies a partial failure of the service.

elapsed_milliseconds The time, in milliseconds, to complete the data service check. This value is reserved for future use.

Return Values The `scds_fm_action()` function returns the following values:

`0` The function succeeded.

nonzero The function failed.

Errors `SCHA_ERR_NOERR` No action was taken, or a restart was successfully attempted.

`SCHA_ERR_FAIL` A failover attempt was made but it did not succeed.

`SCHA_ERR_NOMEM` System is out of memory.

Files `/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_fm_sleep(3HA), scds_initialize(3HA), scha_calls(3HA), scha_control(3HA), scds_fm_print_probes(3HA), scha_resource_setstatus(3HA), `attributes`(5)

Name scds_fm_net_connect – establish a TCP connection to an application

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_fm_net_connect(scds_handle_t handle, scds_socket_t *socklist,
     int count, scds_netaddr_t addr, time_t timeout
```

Description The `scds_fm_net_connect()` function establishes one or more TCP connections (depending on the protocol value of `Port_list` for each address, as described below) to a process that is being monitored.

You can retrieve a list of network addresses for the resource by using scds_get_netaddr_list(3HA). That call also fills the protocol value for each address in the list. If `tcp6` is specified as the protocol in `Port_list` for that address, the protocol value is set to `SCDS_IPPROTO_TCP6`. If `tcp` is specified as the protocol in `Port_list` for that address or if no protocol is specified in `Port_list`, the protocol value is set to `SCDS_IPPROTO_TCP`.

This function also resolves the `hostname` that is supplied in `addr` and connects to:

- The IPv4 address of the `hostname` at the specified port, if the protocol that is specified in `addr` is `SCDS_IPPROTO_TCP`.
- Both the IPv4 address (if there is one) and the IPv6 address (if there is one) of the `hostname` at the specified port, if the protocol specified in `addr` is `SCDS_IPPROTO_TCP6`. The status and the file descriptor, if applicable, are stored in the `scds_socket_t` array that is supplied to this function. The first member of this array is used for the IPv4 mapping and the second member of this array is used for IPv6. The status can be set to one of the following values:
  - `SCDS_FMSOCK_OK` — The operation succeeded and the associated socket file descriptor is valid.
  - `SCDS_FMSOCK_NA` — The address type (IPv4 or IPv6) does not apply to this `hostname`. If the `hostname` contains only one or more IPv4 mappings, the status of the second member in the array that is passed to this function is set to `SCDS_FMSOCK_NA`. The associated socket file descriptor is set to an unknown value, and should never be used.
  - `SCDS_FMSOCK_ERR` — The operation failed or timed out. The associated socket file descriptor is set to an unknown value, and should never be used.

Parameters The following parameters are supported:

handle The handle that is returned by scds_initialize(3HA).

socklist An array of `SCDS_MAX_IPADDR_TYPES` members of type `scds_socket_t`. Each member in the array holds a status and a socket file descriptor for a TCP connection. This parameter is an output argument that is set by this function.

count	The number of members in the *socklist* array. Set this parameter to `SCDS_MAX_IPADDR_TYPES`.
addr	The `hostname`, TCP port number, and protocol identifier that specify where the process is listening.
timeout	The timeout value in seconds. Each socket gets the same time period for a connection to be established before it is timed out. As these time intervals proceed in parallel, this value is effectively the maximum time that the function takes to execute.

Return Values The `scds_fm_net_connect()` function returns the following values:

`0`	The function succeeded. At least one socket connected.
`SCHA_ERR_INVAL`	The function was called with invalid paramaters.
nonzero	Not a single connection could be established, due to a timeout, a refused connection, or some other error. You can inspect the `status` field of all members of the `socklist` array that are set to `SCDS_FMSOCK_ERR` to determine the exact error.

Errors

`SCHA_ERR_NOERR`	Indicates that the function succeeded.
`SCHA_ERR_INTERNAL`	Indicates that an internal error occurred while the function was executing.
`SCHA_ERR_STATE`	Indicates that the connection request was refused by the server.
`SCHA_ERR_TIMEOUT`	Indicates that the connection request timed out.

Examples **EXAMPLE 1** Using the `scds_fm_net_connect()` Function

```
/* this function is called repeatedly,
   after thorough_probe_interval seconds */
int probe(scds_handle_t scds_handle, ...)
{
    scds_socket_t socklist[SCDS_MAX_IPADDR_TYPES];
    ...

    /* for each hostname/port/proto */
    for (i = 0; i < netaddr->num_netaddrs, i++) {
        if (scds_fm_net_connect(scds_handle, socklist,
            SCDS_MAX_IPADDR_TYPES, netaddr[i], timeout) !=
            SCHA_ERR_NOERR)
            {
                /* failed completely */
                ...
        } else {
```

EXAMPLE 1 Using the `scds_fm_net_connect()` Function *(Continued)*

```
            /* at least one sock connected */
            for (j = 0, j < SCDS_MAX_IPADDR_TYPES, j++) {
                if (socklist[j].status == SCDS_FM_SOCK_NA)
                    continue;

                if (socklist[j].status == SCDS_FMSOCK_ERR) {
                    /* this particular connection failed */
                    scds_syslog(LOG_ERR, "Failed: %s",
                        scds_error_string(socklist[j].err));
                    continue;
                }

                /* use socklist[i].fd to perform write/read */
                ...
            }
            (void) scds_fm_net_disconnect(scds_handle, socklist,
                SCDS_MAX_IPADDR_TYPES, remaining_time);
        }

    }
    ...
    return (result);
}
```

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Evolving

See Also scds_fm_net_disconnect(3HA), scds_fm_tcp_connect(3HA), scds_get_netaddr_list(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_fm_net_disconnect – terminate a TCP connection to an application

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t  scds_fm_net_disconnect(scds_handle_t handle, scds_socket_t *socklist,
     int count, time_t timeout
```

Description The `scds_fm_net_disconnect()` function terminates one or more TCP connections to a process that is being monitored.

An attempt is made to close all valid socket connections in the `socklist` array within the specified `timeout` interval. On return, each member of `socklist` contains the value `SCDS_FMSOCK_NA`.

Parameters The following parameters are supported:

handle	The handle that is returned by scds_initialize(3HA).
socklist	The socket list that is returned by scds_fm_net_connect(3HA). This argument is an input/output argument.
count	The number of members in the *socklist* array. Set this parameter to `SCDS_MAX_IPADDR_TYPES`.
timeout	The timeout value in seconds. Each socket gets the same time period to disconnect before it is timed out. As these time intervals proceed in parallel, this value is effectively the maximum time that the function takes to execute.

Return Values The `scds_fm_net_disconnect()` function returns the following values:

`0`	The function succeeded.
`SCHA_ERR_INVAL`	The function was called with invalid paramaters.
Other nonzero values	The function failed. See scha_calls(3HA) for the meaning of failure codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_fm_net_connect(3HA), scds_fm_tcp_disconnect(3HA), scds_initialize(3HA), scha_calls(3HA), attributes(5)

Name scds_fm_print_probes – print probe debugging information

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_fm_print_probes(scds_handle_t handle, int debug_level
```

Description The `scds_fm_print_probes()` function writes probe status information, reported with scds_fm_action(3HA), to the system log. This information includes a list of all probe status history maintained by the DSDL and the timestamp associated with the probe status.

The DSDL defines the maximum debugging level, `SCDS_MAX_DEBUG_LEVEL`, as `9`.

If you specify a `debug_level` greater than the current debugging level being used, no information is written.

Parameters The following parameters are supported:

handle The handle returned from scds_initialize(3HA).

debug_level Debugging level at which the data is to be written. It is an integer between `1` and `SCDS_MAX_DEBUG_LEVEL`, defined as `9` by the DSDL.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_fm_action(3HA), scds_initialize(3HA), scds_syslog_debug(3HA), `attributes`(5)

Name scds_fm_sleep – wait for a message on a fault monitor control socket

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_fm_sleep(scds_handle_t handle, time_t timeout
```

Description The `scds_fm_sleep()` function waits for a data service application process tree that running under control of the process monitor facility to die. If no such death occurs within the specified timeout period, the function returns `SCHA_ERR_NOERR`.

If a data service application process tree death occurs, `scds_fm_sleep()` records `SCDS_COMPLETE_FAILURE` in the failure history and either restarts the process tree or fails it over according to the algorithm described in the scds_fm_action(3HA) man page. If a failover attempt is unsuccessful, a restart of the application is attempted.

If an attempted restart fails, the function returns `SCHA_ERR_INTERNAL`.

Note that if the failure history causes this function to do a failover, and the failover attempt succeeds, `scds_fm_sleep()` never returns.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA).
timeout	The timeout period measured in seconds.

Return Values The `scds_fm_sleep()` function returns the following:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Indicates that the process tree has not died.
`SCHA_ERR_INTERNAL`	Indicates that the data service application process tree has died and failed to restart.
Other values	Indicate the function failed. See scha_calls(3HA) for the meaning of failure codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scha_calls(3HA), scds_fm_action(3HA), scds_initialize(3HA), `attributes`(5)

Name scds_fm_tcp_connect – establish a tcp connection to an application

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_fm_tcp_connect(scds_handle_t handle, int *sock, const char*hostname,
     int port, time_t timeout
```

Description The `scds_fm_tcp_connect()` function establishes a TCP connection with a process being monitored.

Retrieve the hostname with either scds_get_rs_hostnames(3HA) or scds_get_rg_hostnames(3HA).

Consider using scds_fm_net_connect(3HA) instead of this function.

Parameters The following parameters are supported:

handle	The handle returned by scds_initialize(3HA).
sock	A handle to the socket established by this function. This parameter is an output argument set by this function.
hostname	Name of the host where the process is listening. If the *hostname* maps to an IPv4 address only, or to both IPv4 and IPv6 addresses, this function uses the IPv4 mapping as the address at which to connect. If the *hostname* maps to an IPv6 address only, this function uses that IPv6 mapping as the address at which to connect.
port	TCP port number.
timeout	Timeout value in seconds.

Return Values The `scds_fm_tcp_connect()` function returns the following:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Indicates that the function succeeded.
`SCHA_ERR_STATE`	Indicates that an attempt to initiate a connection on a socket failed for reasons other than a timeout.
`SCHA_ERR_TIMEOUT`	Indicates that the function timed out.
Other values	Indicate the function failed. See scha_calls(3HA) for the meaning of failure codes.

Files `/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Deprecated

See Also scds_fm_net_connect(3HA), scds_fm_tcp_disconnect(3HA), scds_get_rg_hostnames(3HA), scds_get_rs_hostnames(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_fm_tcp_disconnect – terminate a tcp connection to an application

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_fm_tcp_disconnect(scds_handle_t handle, int sock, time_t timeout
```

Description The `scds_fm_tcp_disconnect()` function terminates a TCP connection with a process being monitored.

Parameters The following parameters are supported:

handle	The handle returned by scds_initialize(3HA).
sock	The socket number returned by a previous call to scds_fm_tcp_connect(3HA).
timeout	Timeout value in seconds.

Return Values The following exit values are returned:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Indicates that the function succeeded.
`SCHA_ERR_TIMEOUT`	Indicates that the function timed out.
Other values	Indicate that the function failed. See scha_calls(3HA) for the meaning of failure codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Deprecated

See Also scds_fm_net_disconnect(3HA), scds_fm_tcp_connect(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_fm_tcp_read – read data using a tcp connection to an application

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_fm_tcp_read(scds_handle_t  handle, int sock, char *buffer,
     size_t *size, time_t timeout
```

Description The `scds_fm_tcp_read()` function reads data from a TCP connection with a process being monitored.

The *size* argument is an input and argument. On input, you specify the size of the buffer, bytes. On completion, the function places the data in *buffer* and specifies the actual number of bytes read in *size*. If the buffer is not big enough for the number of bytes read, the function returns a full buffer of *size* bytes, and you can call the function again for further data.

If the function times out, it returns `SCHA_ERR_TIMEOUT`. In this case, the function might return fewer bytes than requested, indicated by the value returned in *size*.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA).
sock	The socket number returned by a previous call to scds_fm_tcp_connect(3HA).
buffer	Data buffer.
size	Data buffer size. On input, you specify the size of the buffer, in bytes. On output, the function returns the actual number of bytes read.
timeout	Timeout value in seconds.

Return Values The `scds_fm_tcp_read()` function returns the following:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Indicates that the function succeeded.
`SCHA_ERR_TIMEOUT`	Indicates that the function timed out.
Other values	Indicate that the function failed. See scha_calls(3HA) for the meaning of failure codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_fm_tcp_disconnect(3HA), scds_fm_tcp_write(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_fm_tcp_write – write data using a tcp connection to an application

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t  scds_fm_tcp_write(scds_handle_t handle, int sock, char *buffer,
     size_t *size, time_t timeout
```

Description The `scds_fm_tcp_write()` function writes data from by means of a TCP connection to a process being monitored.

The *size* argument is an input and output argument. On input, you specify the number of bytes to be written. On output, the function returns the number of bytes actually written. If the input and output values of *size* are not equal, an error has occurred. The function returns `SCHA_ERR_TIMEOUT` if it times out before writing all the requested data.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA).
sock	The socket number returned by a previous call to scds_fm_tcp_connect(3HA).
buffer	Data buffer.
size	Data buffer size. On input, you specify the number of bytes to be written. On output, the function returns the number of bytes actually written.
timeout	Timeout value in seconds.

Return Values The `scds_fm_tcp_write()` function returns the following:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Indicates that the function succeeded.
`SCHA_ERR_TIMEOUT`	Indicates that the function timed out.
Other values	Indicate that the function failed. See scha_calls(3HA) for the meaning of failure codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_fm_tcp_connect(3HA), scds_fm_tcp_read(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_free_ext_property – free the resource extension property memory

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_free_ext_property(scha_ext_prop_value_t *property_value
```

Description The `scds_free_ext_property()` function reclaims memory allocated during calls to scds_get_ext_property(3HA).

Parameters The following parameters are supported:

property_value Pointer to a property value.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_get_ext_property(3HA), `attributes`(5)

Name scds_free_netaddr_list – free the network address memory

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_free_netaddr_list(scds_netaddr_list_t *netaddr_list
```

Description The `scds_free_netaddr_list()` function reclaims memory allocated during calls to scds_get_netaddr_list(3HA). It deallocates the memory pointed to by *netaddr_list*.

Parameters The following parameters are supported:

netaddr_list Pointer to a list of hostname-port-protocol 3-tuples used by the resource group.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_get_netaddr_list(3HA), `attributes`(5)

Name scds_free_net_list – free the network resource memory

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_free_net_list(scds_net_resource_list_t *net_resource_list
```

Description The `scds_free_net_list()` function reclaims memory allocated during calls to scds_get_rg_hostnames(3HA) or scds_get_rs_hostnames(3HA). It deallocates the memory pointed to by *netresource_list*.

Parameters The following parameters are supported:

netresource_list Pointer to a list of network resources used by the resource group

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_get_rg_hostnames(3HA), scds_get_rs_hostnames(3HA), `attributes`(5)

Name scds_free_port_list – free the port list memory

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_free_port_list(scds_port_list_t *port_list
```

Description The `scds_free_port_list()` function reclaims memory allocated during calls to scds_get_port_list(3HA). It deallocates the memory pointed to by *port_list*.

Parameters The following parameters are supported:

port_list Pointer to a list of port-protocol pairs used by the resource group

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_get_port_list(3HA), `attributes`(5)

Name scds_get_ext_property – retrieve an extension property

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t  scds_get_ext_property(scds_handle_t handle, const char *property_name,
     scha_prop_type_t  property_type, scha_extprop_value_t **property_value
```

Description The `scds_get_ext_property()` function retrieves the value of a given extension property.

The name of the property is first looked up in the list of properties specified in the method argument list (`argv[]`, which was parsed by `scds_initialize()`). If the property name is not in the method argument list, it is retrieved using the Sun Cluster API. See scha_calls(3HA).

Upon successful completion, the value of the property is placed in the appropriate variable in the union in a `scha_extprop_value_t` structure and a pointer to this structure is passed back to the caller in *property_value*.

You are responsible for freeing memory by using `scds_free_ext_property()`.

You can find information about the data types `scha_prop_type_t` and `scha_extprop_value_t` in scha_calls(3HA) and in the <`scha_types.h`> header file.

DSDL provides convenience functions to retrieve the values of some of the more commonly used resource extension properties. See the scds_property_functions(3HA) man page.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA)
property_name	Name of the property being retrieved
property_type	Property value type. Valid types are defined in scha_calls(3HA) and property_attributes(5).
property_value	Pointer to a property value

Return Values The `scds_get_ext_property()` function returns the following values:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_PROP`	RTR file does not define the specified property.
`SCHA_ERR_NOERR`	The function succeeded.
Other values	Indicate that the function failed. See scha_calls(3HA) for the meaning of the failure codes.

Examples

EXAMPLE 1 Using `scds_get_ext_property()`

```
#include <scha_types.h>
#include <libdsdev.h>
#define INT_EXT_PROP “Int_extension_property”
...
int  retCode;
scha_extprop_value_t *intExtProp;
int retrievedValue;
...
    retCode = scds_get_ext_property(handle,
        INT_EXT_PROP, SCHA_PTYPE_INT, &intExtProp);
    if (retCode != SCHA_ERR_NOERR) {
        scds_syslog(LOG_ERR,
            "Failed to retrieve the extension property %s: %s.",
            INT_EXT_PROP, scds_error_string(retCode));
       ...
    } else {
        retrievedValue = intExtProp->val.val_int;
        ...
        scds_free_ext_property(intExtProp);
        ...
    }
   ...
```

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also

scds_free_ext_property(3HA), scds_initialize(3HA), scds_property_functions(3HA), scha_calls(3HA), rt_reg(4), `attributes`(5), property_attributes(5)

Notes

Only the values of extension properties that are defined in the RTR file can be retrieved by using this function. See rt_reg(4).

Name scds_get_netaddr_list – get the network addresses used by a resource

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>
scha_err_t scds_get_netaddr_list(scds_handle_t handle,
     scds_netaddr_list_t **netaddr_list
```

Description The `scds_get_netaddr_list()` function returns all hostname, port, and protocol combinations that are in use by the resource. These combinations are derived by combining the `Port_list` property settings on the resource with all the hostnames in use by the resource, as returned by the `scds_get_rs_hostnames()` function.

Use `scds_get_netaddr_list()` in a fault monitor to monitor the resource, and to derive the list of hostnames, ports, and protocols that are in use by the resource .

Values for the protocol type are defined in header file <`rgm/libdsdev.h`>.

Free the memory that is allocated and returned by this function with `scds_free_netaddr_list()`.

Parameters The following parameters are supported:

handle	The handle that is returned by `scds_initialize()`
netaddr_list	The list of hostnames, ports, and protocols that are used by the resource group

Return Values The `scds_get_netaddr_list()` function returns the following values:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Indicates that the function succeeded
Other values	Indicate that the function failed. See scha_calls(3HA) for the meaning of failure codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

Attributes See `attributes`(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also scds_free_netaddr_list(3HA), scds_get_rs_hostnames(3HA), scha_calls(3HA), r_properties(5), `attributes`(5)

Name scds_get_port_list – retrieve the port list used by a resource

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_get_port_list(scds_handle_t handle, scds_port_list_t **port_list
```

Description The `scds_get_port_list()` function returns a list of port-protocol pairs used by the resource. Values for the protocol type are defined in the header file `<netinet/in.h>`.

Free the memory allocated and returned by this function with `scds_free_port_list()`.

Parameters The following parameters are supported:

handle The handle returned from `scds_initialize()`

port_list List of port-protocol pairs used by the resource group

Return Values The `scds_get_port_list()` function returns the following:

`0` The function succeeded.

nonzero The function failed.

Errors

`SCHA_ERR_NOERR` Indicates the function succeeded.

Other values Indicate the function failed. See scha_calls(3HA) for the meaning of failure codes.

Files

`/usr/cluster/include/scha.h`
Include file

`/usr/cluster/lib/libscha.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_free_port_list(3HA), scha_calls(3HA), `attributes`(5)

Name scds_get_resource_group_name – retrieve the resource group name

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

 const char *scds_get_resource_group_name(scds_handle_t handle
```

Description The `scds_get_resource_group_name()` function returns a pointer to a character string that is the name of the resource group containing the resource passed to the calling program. The pointer is to memory belonging to the DSDL. Do not modify this memory. A call to `scds_close()` invalidates the pointer.

Parameters The following parameters are supported:

handle The handle returned from `scds_initialize()`

Errors `NULL` Indicates an error condition such as not previously calling scds_initialize(3HA)

See scha_calls(3HA) for a description of other error codes.

Files `/usr/cluster/include/scha.h`
Include file

`/usr/cluster/lib/libscha.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_close(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_get_resource_name – retrieve the resource name

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

const char *scds_get_resource_name(scds_handle_t handle
```

Description The `scds_get_resource_name()` function returns a pointer to a character string containing the name of the resource passed to the calling program. The pointer is to memory belonging to the DSDL. Do not modify this memory. A call to `scds_close()` invalidates the pointer.

Parameters The following parameters are supported:

handle The handle returned from `scds_initialize()`

Errors

`NULL` Indicates an error condition such as not previously calling scds_initialize(3HA)

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_close(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_get_resource_type_name – retrieve the resource type name

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

const char *scds_get_resource_type_name(scds_handle_t handle
```

Description The `scds_get_resource_type_name()` function returns a pointer to a character string containing the name of the resource type of the resource passed to the calling program. The pointer is to memory belonging to the DSDL. Therefore, do not modify this memory. A call to `scds_close()` invalidates the pointer.

Parameters The following parameters are supported:

handle The handle returned from `scds_initialize()`

Errors `NULL` Indicates an error condition such as not previously calling `scds_initialize()`

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_close(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_get_rg_hostnames – get the network resources used in a resource group

Synopsis

```
cc [flags...] -I /usr/cluster/include file
   -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_get_rg_hostnames(char *resourcegroup_name,
scds_net_resource_list_t **netresource_list
```

Description The `scds_get_rg_hostnames()` function retrieves a list of hostnames used by all the network resources in a resource group. This function returns a pointer to the list in *netresource_list*. It is possible for a resource group to contain no network resources or to contain resources that do not use network resources, so this function can return *netresource_list* set to `NULL`.

You can pass the name of any resource group name in the system to `scds_get_rg_hostnames()`. Use the hostnames returned by `scds_get_rg_hostnames()` to contact applications running in the specified resource group.

Free the memory allocated and returned by this function with `scds_free_net_list()`.

Parameters The following parameters are supported

resourcegroup_name	Name of the resource group for which data is to be retrieved
netresource_list	List of network resources used by the resource group

Return Values The `scds_get_rg_hostnames()` function returns the following:

`0`	The function succeeded.
`non-zero`	The function failed.

Errors

`SCHA_ERR_NOERR`	Function succeeded.

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_free_net_list(3HA), scds_get_rs_hostnames(3HA), scha_calls(3HA), `attributes`(5)

Name scds_get_rs_hostnames – get the network resources used by a resource

Synopsis

```
cc [flags...] -I /usr/cluster/include file
   -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_get_rs_hostnames(scds_handle_t handle,
scds_net_resource_list_t **netresource_list
```

Description The `scds_get_rs_hostnames()` function retrieves a list of hostnames used by the resource. If the resource property `Network_resources_used` is set, then the hostnames correspond to the network resources listed in `Network_resources_used`. Otherwise, they correspond to all the network resources in the resource group containing the resource.

This function returns a pointer to the list in *netresource_list*. It is possible for a resource group to contain no network resources or to contain resources that do not use network resources, so this function can return *netresource_list* set to `NULL`.

Free the memory allocated and returned by this function with scds_free_net_list(3HA).

Parameters The following parameters are supported

handle	The handle returned from scds_initialize(3HA)
netresource_list	List of network resources used by the resource group

Return Values The `scds_get_rs_hostnames()` function returns the following:

`0`	The function succeeded
`non-zero`	The function failed

Errors

`SCHA_ERR_NOERR`	Function succeeded.

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_free_net_list(3HA), scds_get_rg_hostnames(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5), r_properties(5)

Name scds_get_zone_name – retrieve the name of a zone on whose behalf a method is running

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

const char *scds_get_zone_name(scds_handle_t handle
```

Description The `scds_get_zone_name()` function returns a pointer to a character string. If the following conditions are met, this character string contains the name of the zone in which a resource group runs:

- The `Global_zone` resource type property is set to `TRUE`.

  See the rt_properties(5) man page for information about the `Global_zone` resource type property.
- The method is configured to run in a non-global zone.

In all other cases, including the following, the character string is NULL:

- Sun Cluster software is running on an operating system that does not support zones.
- The resource group and the method are running in the global zone.
- The `Global_zone` resource type property is set to `FALSE`.

To obtain the name of the zone in which a method is actually executing, use the `zonename` command. See the `zonename`(1) man page.

The pointer to the character string points to memory that belongs to the Data Service Development Library (DSDL). Do not modify this memory. A call to `scds_close()` invalidates this pointer.

Parameters The following parameters are supported:

handle The handle that is returned from scds_initialize(3HA).

Errors

`SCHA_ERR_NOERR` The function succeeded.

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also zonename(1), scds_close(3HA), scds_initialize(3HA), scha_calls(3HA), attributes(5), rt_properties(5)

Name scds_hasp_check – get status information about SUNW.HAStoragePlus resources that are used by a resource

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_hasp_check(scds_handle_t handle, scds_hasp_status_t *hasp_status
```

Description The `scds_hasp_check()` function retrieves status information about SUNW.HAStoragePlus(5) resources that are used by a resource. This information is obtained from the state, online or otherwise, of all `SUNW.HAStoragePlus` resources on which the resource depends. This state is obtained by using the `Resource_dependencies` or `Resource_dependencies_weak` system properties that are defined for the resource.

Resource type implementations can use `scds_hasp_check()` in `VALIDATE` and `MONITOR_CHECK` method callback implementations to determine whether checks that are specific to any file systems that are managed by `SUNW.HAStoragePlus` resources should be carried out.

When the function succeeds, a status code is stored in *hasp_status*. This code can be one of the following values:

`SCDS_HASP_NO_RESOURCE`
: Indicates that the resource does not depend on a `SUNW.HAStoragePlus` resource.

`SCDS_HASP_ERR_CONFIG`
: Indicates that at least one of the `SUNW.HAStoragePlus` resources on which the resource depends is located in a different resource group.

`SCDS_HASP_NOT_ONLINE`
: Indicates that a `SUNW.HAStoragePlus` resource on which the resource depends is not online on any potential primary node or zone.

`SCDS_HASP_ONLINE_NOT_LOCAL`
: Indicates that at least one `SUNW.HAStoragePlus` resource on which the resource depends is online, but on another node or zone from which this function is called.

`SCDS_HASP_ONLINE_LOCAL`
: Indicates that all `SUNW.HAStoragePlus` resources on which the resource depends are online on the node or zone from which this function is called.

Note – The preceding status codes have precedence over each other in the order in which they appear. For example, if a `SUNW.HAStoragePlus` resource is not online and another `SUNW.HAStoragePlus` resource is online on a different node or zone, the status code is set to `SCDS_HASP_NOT_ONLINE` rather than `SCDS_HASP_ONLINE_NOT_LOCAL`.

The `scds_hasp_check()` function ignores all `SUNW.HAStoragePlus` resources for which the `FilesystemMountPoints` or `Zpools` property is set to an empty list, the default.

Parameters The following parameters are supported:

handle Handle that is returned from scds_initialize(3HA).

hasp_status Status of `SUNW.HAStoragePlus` resources that are used by the resource.

Return Values

`SCHA_ERR_NOERR` The function succeeded.

This value also indicates that the status code that is stored in `hasp_status` is valid.

`SCHA_ERR_INTERNAL` The function failed.

The value that is stored in `hasp_status` is undefined. Ignore this undefined value.

See the scha_calls(3HA) man page for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also

scds_initialize(3HA), scha_calls(3HA), `attributes`(5), SUNW.HAStoragePlus(5)

Name scds_initialize – allocate and initialize DSDL environment

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_initialize(scds_handle_t *handleint argc, char *argv[]
```

Description The `scds_initialize()` function initializes the DSDL environment. You must call this function once at the beginning of each program or fault monitor that uses any other DSDL functions.

The `scds_initialize()` function does the following:

- Checks and processes the command line arguments (*argc* and *argv[]*) that the framework passes to the calling program and that must be passed along to `scds_initialize()`. No further processing of the command line arguments is required of the calling program. See EXAMPLES.
- Sets up internal data structures with information needed by the other functions in the DSDL. It retrieves resource, resource type, and resource group property values and stores them in these data structures. Values for any properties supplied on the command line by means of the *argv[]* argument take precedence over those retrieved from the RGM. That is, if a new value for a property has been specified in the command line arguments *(argv[])* passed to the data service method, then this new value is returned by the function that retrieves that property's value. Otherwise, the existing value retrieved from the RGM is returned.
- Initializes the data service fault monitoring information
- Initializes the logging environment. All syslog messages are prefixed with: `SC[`*<resourceTypeName>*`,`*<resourceGroupName>*`,`*<resourceName>*`,`*<methodName>*

  Functions that send messages to syslog use the facility returned by `scha_cluster_getlogfacility()`. These messages can be forwarded to appropriate log files and users. See `syslog.conf`(4) for more information.
- Validates fault monitor probe settings. It verifies that the `Retry_interval` is greater than or equal to (`Thorough_probe_interval` * `Retry_count`). If this is not true, it sends an appropriate message to the syslog facility. You could call `scds_initialize()` and `scds_close()` in a `VALIDATE` method for this validation of the fault monitor probe settings even if you call no other DSDL functions in the `VALIDATE` method.

If `scds_initialize()` succeeds, you must call `scds_close()` before exiting the calling program.

If `scds_initialize()` fails, you must not call `scds_close()` to clean up. When `scds_initialize()` fails, do not call any other DSDL functions. Otherwise, they return `SCHA_ERR_INVAL` or a `NULL` value. Instead, call `exit()` with a non-zero argument.

Parameters The following parameters are supported:

handle	A handle initialized by `scds_initialize()` and used by other DSDL functions.
argc	Number of arguments that is passed to the calling program.
argv	Pointer to an argument array passed to the calling program.

Errors

`SCHA_ERR_NOERR`	The function succeeded.

See scha_calls(3HA) for a description of other error codes.

Examples

EXAMPLE 1 Using `scds_initialize()`

```
int main(int argc, char *argv[]){
 scds_handle_t handle;

 if (scds_initialize(&handle, argc, argv) !=
 SCHA_ERR_NOERR)
 exit(1);
 ...
 /* data service code */
...
 scds_close(&handle);
}
```

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also

scds_close(3HA), scds_property_functions(3HA), scha_calls(3HA), scha_cluster_getlogfacility(3HA), `syslog.conf`(4), r_properties(5)

Name scds_pmf_get_status – determine if a PMF-monitored process tree exists

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_pmf_get_status(scds_handle_t handle, scds_pmf_type_t program_type,
     int instance,  scds_pmf_status_t*pmf_status
```

Description The `scds_pmf_get_status()` function determines if the specified instance is being monitored under PMF control. This function is equivalent to the pmfadm(1M) command with the `-q` option.

Parameters The following parameters are supported:

handle The handle returned from `scds_initialize()`

program_type Type of program to execute. Valid types are:

`SCDS_PMF_TYPE_SVC`	Data service application
`SCDS_PMF_TYPE_MON`	Fault monitor
`SCDS_PMF_TYPE_OTHER`	Other

instance For resources with multiple instances, this integer, starting at 0, uniquely identifies the instance. For single instance resources, use 0.

pmf_status If PMF is monitoring the specified instance, *pmf_status* is set to `SCDS_PMF_MONITORED`. Otherwise it is set to `SCDS_PMF_NOT_MONITORED`.

Return Values The `scds_pmf_get_status()` function returns the following:

`0`	The function succeeded.
`non-zero`	The function failed.

Errors

`SCHA_ERR_NOERR`	Function succeeded

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also pmfadm(1M), scds_initialize(3HA), scha_calls(3HA), `attributes`(5)

Name scds_pmf_restart_fm – restart fault monitor using PMF

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_pmf_restart_fm(scds_handle_t handle, int instance
```

Description The `scds_pmf_restart_fm()` function sends a `SIGKILL` signal to the fault monitor process tree to kill the fault monitor and then uses PMF to restart it. This function uses the `MONITOR_STOP_TIMEOUT` property as its timeout value. That is, `scds_pmf_restart_fm()` waits at most the value of the `MONITOR_STOP_TIMEOUT` property for the process tree to die.

If the `MONITOR_STOP_TIMEOUT` property is not explicitly set in the RTR file, the default timeout value is used.

One way to use this function is to call it in an `UPDATE` method to restart the monitor, possibly with new parameters.

Parameters The following parameters are supported:

handle The handle returned from `scds_initialize()`

instance For resources with multiple instances of the fault monitor, this integer, starting at 0, uniquely identifies the fault monitor instance. For single instance fault monitors, use 0.

Return Values The `scds_pmf_restart_fm()` function returns the following:

`0` The function succeeded.

`non-zero` The function failed.

Errors `SCHA_ERR_NOERR` Function succeeded

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also pmfadm(1M), scha_calls(3HA), `signal`(3HEAD), `attributes`(5) , r_properties(5)

Name scds_pmf_signal – send a signal to a process tree under PMF control

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_pmf_signal(scds_handle_t handle, scds_pmf_type_t program_type,
     int instance, int signal, time_t timeout
```

Description The `scds_pmf_signal()` function sends the specified signal to a process tree running under PMF control. This function is equivalent to the pmfadm(1M) command with the `-k` option.

After sending the signal, the `scds_pmf_signal()` function waits for the specified timeout period for the process tree to die, before returning. A value of `0` for `timeout` tells the function to return immediately without waiting for any process to exit. A value of `-1` tells the function to wait indefintely for the processes to exit.

Parameters The following parameters are supported:

handle	The handle returned from `scds_initialize()`	
program_type	Type of program to execute. Valid types are:	
	`SCDS_PMF_TYPE_SVC`	Data service application
	`SCDS_PMF_TYPE_MON`	Fault monitor
	`SCDS_PMF_TYPE_OTHER`	Other
instance	For resources with multiple instances, this integer, starting at 0, uniquely identifies the instance. For single instance resources, use 0.	
signal	Solaris signal to send. See `signal`(3HEAD).	
timeout	Timeout period in seconds.	

Return Values The `scds_pmf_signal()` function returns the following:

`0`	The function succeeded.
`non-zero`	The function failed.

Errors

`SCHA_ERR_TIMEOUT`	The process tree did not exit within the specified timeout period after the signal was sent.
`SCHA_ERR_NOERR`	The function succeeded.
Other values	Indicate the function failed. See scha_calls(3HA) for the meaning of failure codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also pmfadm(1M), scds_initialize(3HA), scha_calls(3HA), `signal`(3HEAD), `attributes`(5)

Name scds_pmf_start, scds_pmf_start_env – execute a program under PMF control

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_pmf_start(scds_handle_t handle, scds_pmf_type_t program_type,
     int instance, const char *command, int child_monitor_level)

scha_err_t scds_pmf_start_env(scds_handle_t handle, scds_pmf_type_t program_type,
     int instance, const char *command, int child_monitor_level, char ** env)
```

Description The `scds_pmf_start()` function executes a program, specified by *command*, under PMF control. This function is equivalent to the pmfadm(1M) command with the `-c` option.

The *command* argument contains a command line and command line arguments that are passed to the function.

When you start a data service application or other process (program type `SCDS_PMF_TYPE_SVC` or `SCDS_PMF_TYPE_OTHER`) under PMF with `scds_pmf_start()`, you choose the level of child processes to monitor by using the `child_monitor_level` argument. Values for the `child_monitor_level` argument are `none`, `some` or `all`. The `child_monitor_level` argument specifies that children up to and including level `child_monitor_level` will be monitored. The original process is executed at level 0, its children at level 1, their children at level 2, and so on. Any new fork operation produces a new level of children. Specify `-1` to monitor all levels of children.

For example, if the command to start is a daemon, the appropriate `child_monitor_level` is `1`. If the command to start is a script that starts a daemon, the appropriate value for `child_monitor_level` is `2`.

For a fault monitor (program type `SCDS_PMF_TYPE_MON`), the `child_monitor_level` argument is ignored and `0` is used.

If the underlying application process is already running, `scds_pmf_start()` prints a `syslog()` error and returns `SCHA_ERR_INTERNAL` because the RGM guarantees that two calls to a `START` function on a node must have an intervening `STOP` function.

Parameters The following parameters are supported:

handle The handle returned from scds_initialize(3HA)

program_type Type of program to execute. Valid types are:

`SCDS_PMF_TYPE_SVC`	Data service application
`SCDS_PMF_TYPE_MON`	Fault monitor
`SCDS_PMF_TYPE_OTHER`	Other

instance For resources with multiple instances, this integer, starting at 0, uniquely identifies the instance. For single instance resources, use 0.

command Command, including command line arguments, to execute under PMF control.

child_monitor_level For `program_type` `SCDS_PMF_TYPE_SVC` and `SCDS_PMF_TYPE_OTHER`, this argument specifies the level of child processes to be monitored (equivalent to the `-C` option to `pmfadm`). Use `-1` to specify all levels of child processes. For `program_type` `SCDS_PMF_TYPE_MON`, this argument is ignored.

env Specifies an array of character pointers to environment strings, which are described in the `execve`(2) man page. When the program that the *command* parameter specifies is executed, this environment is passed to this program.

Return Values The `scds_pmf_start()` function returns the following:

`0` The function succeeded.

`non-zero` The function failed.

Errors

`SCHA_ERR_INTERNAL`	The underlying application process is already running.
`SCHA_ERR_NOERR`	The function succeeded.
Other values	The function failed. See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also pmfadm(1M), scds_initialize(3HA), scds_pmf_stop(3HA), scds_svc_wait(3HA), scha_calls(3HA), `execve`(2), `attributes`(5)

Name scds_pmf_stop – terminate a process that is running under PMF control

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t  scds_pmf_stop(scds_handle_t handle, scds_pmf_type_t program_type,
     int instance, int signal, time_t timeout
```

Description The `scds_pmf_stop()` function stops a program that is running under PMF control. It is equivalent to the pmfadm(1M) command with the `-s` option.

If the requested instance is not running, `scds_pmf_stop()` returns with value `SCHA_ERR_NOERR`.

If the requested instance is running, then the specified signal is sent to the instance. If the instance fails to die within a period of time equal to 80% of the timeout value, `SIGKILL` is sent to the instance. If the instance then fails to die within a period of time equal to 15% of the timeout value, the function is considered to have failed and returns `SCHA_ERR_TIMEOUT`. The remaining 5% of the timeout argument is presumed to have been absorbed by this function's overhead.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA)	
program_type	Type of program to execute. Valid types are:	
	`SCDS_PMF_TYPE_SVC`	Data service application
	`SCDS_PMF_TYPE_MON`	Fault monitor
	`SCDS_PMF_TYPE_OTHER`	Other
instance	For resources with multiple instances, this integer, starting at 0, uniquely identifies the instance. For single instance resources, use 0.	
signal	Solaris signal to send kill the instance. See `signal`(3HEAD). Use `SIGKILL` if the specified signal fails to kill the instance.	
timeout	Timeout period measured in seconds.	

Return Values The `scds_pmf_stop()` function returns the following:

`0`	The function succeeded.
`non-zero`	The function failed.

Errors

`SCHA_ERR_TIMEOUT`	The function timed out.
`SCHA_ERR_NOERR`	The function succeeded.
Other values	Indicate the function failed. See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also pmfadm(1M), scds_initialize(3HA), scds_pmf_start(3HA), scha_calls(3HA), `signal`(3HEAD), `attributes`(5)

Name scds_pmf_stop_monitoring – stop monitoring a process that is running under PMF control

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t  scds_pmf_stop_monitoring(scds_handle_t handle,
     scds_pmf_type_t program_type,
     int instance
```

Description The `scds_pmf_stop_monitoring()` function stops the monitoring of a process tree that is running under PMF control. PMF does not send a signal to stop the process. Rather, PMF makes no future attempts to restart the process.

If the requested process is not under PMF control, `scds_pmf_stop_monitoring()` returns, with value `SCHA_ERR_NOERR`.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA)	
program_type	Type of program to execute. Valid types are:	
	`SCDS_PMF_TYPE_SVC`	Data service application
	`SCDS_PMF_TYPE_MON`	Fault monitor
	`SCDS_PMF_TYPE_OTHER`	Other
instance	For resources with multiple instances, this integer, starting at 0, uniquely identifies the instance. For single instance resources, use 0.	

Return Values The `scds_pmf_stop_monitoring()` function returns the following:

`0`	The function succeeded.
`non-zero`	The function failed.

Errors

`SCHA_ERR_NOERR`	The function succeeded.

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also pmfadm(1M), scds_initialize(3HA), scds_pmf_start(3HA), scds_pmf_stop(3HA), scha_calls(3HA), `attributes`(5)

Name scds_print_netaddr_list – print the contents of a list of hostname-port-protocol 3-tuples used by a resource group

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_print_netaddr_list(scds_handle_t handle, int debug_level,
     constscds_netaddr_list_t *netaddr_list
```

Description The `scds_print_netaddr_list()` function writes the contents of a list of hostname-port-protocol 3-tuples, pointed to by *netaddr_list*, to the system log, at the debugging level specified by *debug_level*. If the specified debugging level is greater than the debugging level currently being used, no information is written.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA)
debug_level	The debugging level at which the data is to be written
netaddr_list	Pointer to a list of hostname-port-protocol 3-tuples used by the resource group, retrieved with scds_get_netaddr_list(3HA)

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_get_netaddr_list(3HA), scds_initialize(3HA), scds_syslog_debug(3HA), `attributes`(5)

Name scds_print_net_list – print the contents of a network resource list

Synopsis

```
cc [flags...] -I /usr/cluster/include file
   -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_print_net_list(scds_handle_t handle,
int debug_level,
constscds_net_resource_list_t *netresource_list
```

Description The `scds_print_net_list()` function writes the contents of the network resource list, pointed to by *netresource_list*, to the system log, at the debugging level specified by *debug_level*. If the specified debugging level is greater than the debugging level currently being used, no information is written.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA)
debug_level	Debugging level at which the data is to be written
netresource_list	Pointer to an initialized network resource list, retrieved with either scds_get_rg_hostnames(3HA) or scds_get_rs_hostnames(3HA)

Files `/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_get_rg_hostnames(3HA), scds_get_rs_hostnames(3HA), scds_initialize(3HA), scds_syslog_debug(3HA), `attributes`(5)

Name scds_print_port_list – print the contents of a port list

Synopsis

```
cc [flags...] -I /usr/cluster/include file
   -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_print_port_list(scds_handle_t handle,
int debug_level, constscds_port_list_t *port_list
```

Description The `scds_print_port_list()` function writes the contents of a port list, pointed to by *port_list*, to the system log, at the debugging level specified by *debug_level*. If the specified debugging level is greater than the debugging level currently being used, no information is written.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA)
debug_level	Debugging level at which the data is to be written
port_list	Pointer to a list of port-protocol pairs used by the resource group, retrieved with `scds_get_port_list()`.

Files `/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_get_port_list(3HA), scds_initialize(3HA), scds_syslog_debug(3HA), `attributes`(5)

Name scds_property_functions – A set of convenience functions to retrieve values of commonly used resource properties, resource group properties, resource type properties, and extension properties

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

return_value_type scds_get_property_name(scds_handle_t handle);
```

Description The Data Service Development Library (DSDL) provides a set of convenience functions to retrieve values of commonly used resource properties, resource group properties, resource type properties, and extension properties. Retrieve user-defined extension properties with scds_get_ext_property(3HA).

All convenience functions use the following conventions:

- The functions take only the *handle* argument. The *handle* argument to be passed to the property retrieval function is returned by a prior call to scds_initialize(3HA).
- Each function corresponds to a particular property.
- The return value type of the function matches the type of the property value that the function retrieves.
- These functions do not return errors because the return values have been pre-computed in scds_initialize(3HA). For functions that return pointers, a `NULL` value is returned when an error condition is encountered, for example, when `scds_initialize()` was not previously called.
- If a new value for a property has been specified in the command-line arguments that are passed to the calling program (*argv[]*), this new value is returned (in the case of the implementation of a Validate method). By this means, you can validate prospective new property values before they are actually set. Otherwise, these functions return the value that is retrieved from the RGM.
- Some of these convenience functions return a pointer to memory belonging to the DSDL. Do not modify this memory. A call to scds_close(3HA) invalidates this pointer.

See the r_properties(5), rg_properties(5), and rt_properties(5) man pages for descriptions of standard properties. See the individual data service man pages for descriptions of extension properties.

See the scha_calls(3HA) man page and the `scha_types.h`header file for information about the data types used by these functions, such as `scha_prop_type_t`, `scha_extprop_value_t`, `scha_initnodes_flag_t`, `scha_str_array_t`, `scha_failover_mode_t`, `scha_switch_t`, and `scha_rsstatus_t`.

These functions use the following naming conventions:

Resource property
`scds_get_rs_`*property-name*

Resource group property
`scds_get_rg_`*property-name*

Resource type property
`scds_get_rt_`*property-name*

Commonly used extension property
`scds_get_ext_`*property-name*

Note – Property names are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify property names.

Resource-Specific Functions

The function returns the value of a specific resource property. Some of the properties' values are explicitly set either in the RTR file or by a clresource(1CL) command. Others are determined dynamically by the RGM. The functions return data types that correspond to the requested property.

`Cheap_probe_interval`
```
int scds_get_rs_cheap_probe_interval(scds_handle_t handle)
```

`Failover_mode`
```
scha_failover_mode_t scds_get_rs_failover_mode(scds_handle_t handle)
```

`Monitor_stop_timeout`
```
int scds_get_rs_monitor_stop_timeout(scds_handle_t handle)
```

`Monitored_switch`
```
scha_switch_t scds_get_rs_monitored_switch(scds_handle_t handle)
```

`Network_resources_used`
```
scha_str_array_t * scds_get_rs_network_resources_used(scds_handle_t handle)
```

`On_off_switch`
```
scha_switch_t scds_get_rs_on_off_switch(scds_handle_t handle)
```

`Resource_dependencies`
```
const scha_str_array_t * scds_get_rs_resource_dependencies(scds_handle_t
handle)
```

`Resource_dependencies_Q`
```
const scha_str_array_t * scds_get_rs_resource_dependencies_Q(scds_handle_t
handle)
```

`Resource_dependencies_offline_restart`
```
const scha_str_array_t *
scds_get_rs_resource_dependencies_offline_restart(scds_handle_t handle)
```

`Resource_dependencies_Q_offline_restart`
```
const scha_str_array_t *
scds_get_rs_resource_dependencies_Q_offline_restart(scds_handle_t handle)
```

```
Resource_dependencies_restart
   const scha_str_array_t *
   scds_get_rs_resource_dependencies_restart(scds_handle_t handle)

Resource_dependencies_Q_restart
   const scha_str_array_t *
   scds_get_rs_resource_dependencies_Q_restart(scds_handle_t handle)

Resource_dependencies_weak
   const scha_str_array_t *
   scds_get_rs_resource_dependencies_weak(scds_handle_t handle)

Resource_dependencies_Q_weak
   const scha_str_array_t *
   scds_get_rs_resource_dependencies_Q_weak(scds_handle_t handle)

Resource_project_name
   const char * scds_get_rs_resource_project_name(scds_handle_t handle)

Retry_count
   int scds_get_rs_retry_count(scds_handle_t handle)

Retry_interval
   int scds_get_rs_retry_interval(scds_handle_t handle)

Scalable
   boolean scds_get_rs_scalable(scds_handle_t handle)

Start_timeout
   int scds_get_rs_start_timeout(scds_handle_t handle)

Stop_timeout
   int scds_get_rs_stop_timeout(scds_handle_t handle)

Thorough_probe_interval
   int scds_get_rs_thorough_probe_interval(scds_handle_t handle)
```

Resource Group-Specific Functions

The function returns the value of a specific resource group property. Some of the properties' values are explicitly set by a clresourcegroup(1CL) command. Others are determined dynamically by the RGM. The functions return data types appropriate for the requested property.

```
Desired_primaries
   int scds_get_rg_desired_primaries(scds_handle_t handle)

Global_resources_used
   const scha_str_array_t * scds_get_rg_global_resources_used(scds_handle_t
   handle)

Implicit_network_dependencies
   boolean_t scds_get_rg_implicit_network_dependencies(scds_handle_t handle)
```

```
Maximum_primaries
   int scds_get_rg_maximum_primaries(scds_handle_t handle)

Nodelist
   const scha_str_array_t * scds_get_rg_nodelist (scds_handle_t handle)

Pathprefix
   const char * scds_get_rg_pathprefix(scds_handle_t handle)

Pingpong_interval
   int scds_get_rg_pingpong_interval(scds_handle_t handle)

Resource_list
   const scha_str_array_t * scds_get_rg_resource_list(scds_handle_t handle)

RG_affinities
   const scha_str_array_t * scds_get_rg_rg_affinities(scds_handle_t handle)

RG_mode
   scha_rgmode_t scds_get_rg_rg_mode(scds_handle_t handle)

RG_project_name
   const char * scds_get_rg_rg_project_name(scds_handle_t handle)

RG_slm_cpu_shares
   int scds_get_rg_rg_slm_cpu_shares(scds_handle_t handle)

RG_slm_pset_min
   int scds_get_rg_rg_slm_pset_min(scds_handle_t handle)

RG_slm_pset_type
   const char * scds_get_rg_rg_slm_pset_type(scds_handle_t handle)

RG_slm_type
   const char * scds_get_rg_rg_slm_type(scds_handle_t handle)
```

Resource Type-Specific Functions

The function returns the value of a specific resource type property. Some of the properties' values are explicitly set either in the RTR file or by a clresourcetype(1CL) command. Others are determined dynamically by the RGM. The functions return data types appropriate for the requested property.

```
API_version
   int scds_get_rt_api_version(scds_handle_t handle)

Failover
   boolean_t scds_get_rt_failover(scds_handle_t handle)

Init_nodes
   scha_initnodes_flag_t scds_get_rt_init_nodes(scds_handle_t handle)

Installed_nodes
   const scha_str_array_t * scds_get_rt_installed_nodes(scds_handle_t handle)
```

```
RT_basedir
   const char * scds_get_rt_rt_basedir(scds_handle_t handle)

RT_version
   const char * scds_get_rt_rt_version(scds_handle_t handle)

Single_instance
   boolean_t scds_get_rt_single_instance(scds_handle_t handle)

Start_method
   const char * scds_get_rt_start_method(scds_handle_t handle)

Stop_method
   const char * scds_get_rt_stop_method(scds_handle_t handle)
```

Extension Property-Specific Functions

The function returns the value of a specific resource extension property. The properties' values are explicitly set either in the RTR file or by a clresource(1CL) command. The functions return data types appropriate for the requested property.

A resource type can define extension properties beyond the four listed here, but these four properties have convenience functions defined for them. You retrieve these properties with these convenience functions or with the scds_get_ext_property(3HA) function. You must use `scds_get_ext_property()` to retrieve extension properties other than these four.

```
Confdir_list
   scha_str_array_t * scds_get_ext_confdir_list(scds_handle_t handle)

Monitor_retry_count
   int scds_get_ext_monitor_retry_count(scds_handle_t handle)

Monitor_retry_interval
   int scds_get_ext_monitor_retry_interval(scds_handle_t handle)

Probe_timeout
   int scds_get_ext_probe_timeout(scds_handle_t handle)
```

Parameters

The following parameter is supported for all the convenience functions:

handle The handle that is returned from scds_initialize(3HA).

Return Values

The return value type of the function matches the type of the property value that the function retrieves.

These functions do not return errors because the return values have been pre-computed in scds_initialize(3HA). For functions that return pointers, a `NULL` value is returned when an error condition is encountered, for example, when `scds_initialize()` was not previously called.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also clresource(1CL), clresourcegroup(1CL), clresourcetype(1CL), scds_close(3HA), scds_get_ext_property(3HA), scds_get_port_list(3HA), scds_get_resource_group_name(3HA), scds_get_resource_name(3HA), scds_get_resource_type_name(3HA), scds_initialize(3HA), scha_calls(3HA), `attributes`(5), r_properties(5), rg_properties(5), and rt_properties(5)

Name scds_restart_resource – restart a resource

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_restart_resource(scds_handle_t handle
```

Description The `scds_restart_resource()` function provides resource-level granularity for the restart operation. This function calls the `STOP` method and then the `START` method for the resource passed to the calling program. If `PRENET_START` and `POSTNET_STOP` methods are defined for the resource type, they are ignored. Call this function from the fault monitor.

Parameters The following parameters are supported:

handle The handle returned from scds_initialize(3HA)

Return Values The `scha_restart_resource()` function returns the following:

`0` The function succeeded.

`non-zero` The function failed.

Errors `SCHA_ERR_NOERR` Function succeeded.

See scha_calls(3HA) for a description of other error codes.

Files `/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also rt_callbacks(1HA), scds_restart_rg(3HA), scha_calls(3HA), scha_control(3HA), `attributes`(5)

Name scds_restart_rg – restart a resource group

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_restart_rg(scds_handle_t handle
```

Description The `scds_restart_rg()` function performs an scha_control(3HA) `SCHA_RESTART` operation on the resource group containing the resource passed to the calling program. Call this function from the fault monitor.

When this function succeeds, it does not return. Therefore, treat this function as the last piece of code to be executed in the calling program.

Parameters The following parameters are supported:

handle The handle returned from scds_initialize(3HA)

Return Values The `scds_restart_rg()` function returns the following:

`0` The function succeeded.

`non-zero` The function failed.

Errors `SCHA_ERR_NOERR` Function succeeded.

See scha_calls(3HA) for a description of other error codes.

Files `/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scha_calls(3HA), scha_control(3HA), scds_initialize(3HA), scds_restart_resource(3HA), `attributes`(5)

Name scds_simple_net_probe – probe by establishing and terminating a TCP connection to an application

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_simple_net_probe(scds_handle_t handle, scds_netaddr_t addr,
     time_t timeout, scds_fmsock_status_t *status, int count
```

Description The `scds_simple_net_probe()` function is a wrapper function around scds_fm_net_connect(3HA) and scds_fm_net_disconnect(3HA). For hosts that have multiple mappings, `scds_simple_net_probe()` handles both IPv4 and IPv6 addresses for the supplied `hostname`.

You can retrieve a list of network addresses for the resource by using scds_get_netaddr_list(3HA).

The status for a connect to, or disconnect from, an IPv4 target is stored in the first member of the `scds_fmsock_status_t` array. The second member contains the status for an IPv6 target. If the `hostname` that is supplied to this function does not contain an IPv4 or IPv6 mapping, the corresponding status is set to `SCDS_FMSOCK_NA`.

Parameters The following parameters are supported:

handle	The handle returned by scds_initialize(3HA).
addr	The `hostname`, TCP port number, and protocol identifier that specify where the process is listening.
timeout	The timeout value in seconds to wait for a successful connection. Each socket (IPv4 or IPv6) gets the same timeout period, and timeouts proceed in parallel.
status	Array of `SCDS_MAX_IPADDR_TYPES` members of type `scds_fmsock_status_t`. Each member in the array holds a status. This parameter is an output argument that is set by this function.
count	The number of members in the *socklist* array. Set this parameter to `SCDS_MAX_IPADDR_TYPES`.

Return Values The `scds_simple_net_probe()` function returns the following values:

`0`	The function succeeded.
`SCHA_ERR_INVAL`	The function was called with invalid paramaters.
Other nonzero values	At least one connect operation failed due to a timeout, a refused connection, or some other error. Inspect the `err` field of all members of the `socklist` array that are set to `SCDS_FMSOCK_ERR` to determine the exact error.

nonzero	At least one connect or disconnect operation failed. You can inspect the `scds_fmsock_status_t` array to determine if the failure was in an IPv4 target, an IPv6 target, or both.

Errors

`SCHA_ERR_NOERR`	Indicates that the function succeeded.
`SCHA_ERR_INTERNAL`	Indicates that an internal error occurred while the function was executing.
`SCHA_ERR_STATE`	Indicates that the connection request was refused by the server.
`SCHA_ERR_TIMEOUT`	Indicates that the connection request timed out.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Evolving

See Also scds_fm_net_connect(3HA), scds_fm_net_disconnect(3HA), scds_get_netaddr_list(3HA), scds_initialize(3HA), scds_simple_probe(3HA), scha_calls(3HA), `attributes`(5)

Name scds_simple_probe – probe by establishing and terminating a TCP connection to an application

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_simple_probe(scds_handle_t handle, const char *hostname,
     int port, time_t timeout
```

Description The `scds_simple_probe()` function is a wrapper function around `connect`(3SOCKET) and `close`(2) to run under a timeout.

Retrieve the *hostname* with either scds_get_rg_hostnames(3HA) or scds_get_rs_hostnames(3HA).

Consider using scds_simple_net_probe(3HA) instead of this function.

Parameters The following parameters are supported:

handle	The handle returned by scds_initialize(3HA).
hostname	Internet hostname of the machine to which to connect.
port	Port number with which to make the connection.
timeout	Timeout value in seconds (to wait for a successful connection).

Return Values The `scds_simple_probe()` function returns the following:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Indicates that the function succeeded.
`SCHA_ERR_TIMEOUT`	Indicates that the function timed out.

See scha_calls(3HA) for a description of other error codes.

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Deprecated

See Also `close`(2), `connect`(3SOCKET), scds_fm_net_connect(3HA), scds_fm_net_disconnect(3HA), scds_get_rg_hostnames(3HA), scds_get_rs_hostnames(3HA), scds_initialize(3HA), scds_simple_net_probe(3HA), scha_calls(3HA), `attributes`(5)

Name scds_svc_wait – wait for the specified timeout period for a monitored process to die

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_svc_wait(scds_handle_t handle, time_t timeout
```

Description The `scds_svc_wait()` function waits for the specified timeout period for a monitored process group to die. It waits upon all process groups started by scds_pmf_start(3HA) for the resource passed to the calling `START` method. The `scds_svc_wait()` function uses the `Retry_interval` and `Retry_count` properties of the resource to limit the number of process deaths to wait on. If the number of process deaths during `Retry_interval` reaches the value of `Retry_count`, `scds_svc_wait()` returns with `SCHA_ERR_FAIL`.

If the number of process failures is below the value of `Retry_count`, the process is restarted and `scds_svc_wait()` waits the full timeout period for further process deaths. The counting of process failures spans successive calls to `scds_svc_wait()`.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA)
timeout	Timeout period measured in seconds

Return Values The `scds_svc_wait()` function returns the following:

`0`	The function succeeded.
`non-zero`	The function failed.

Errors

`SCHA_ERR_TIMEOUT`	The function timed out.
`SCHA_ERR_NOERR`	No process deaths occurred, or a process was successfully restarted.
`SCHA_ERR_FAIL`	The number of failures reached the value of the `Retry_count` property.
`SCHA_ERR_STATE`	A system error or an otherwise unexpected error occurred.

See scha_calls(3HA) for a description of other error codes.

Examples **EXAMPLE 1** Using `scds_svc_wait()` in a `START` Method

The following example shows how you could use `scds_svc_wait` in a `START` method to return early if the service fails to start. After starting an application process with `scds_pmf_start()`, a `START` method must wait for the application to fully initialize itself and become available before returning success. If the application fails to start, the `START` method must wait the entire `Start_timeout` period before returning with failure. Using `scds_svc_wait()`, as in the following example, allows `START` methods to restart applications up to `Retry_count` times and return early with failure from the `START` method if the service is unable to start up.

EXAMPLE 1 Using `scds_svc_wait()` in a START Method *(Continued)*

```
/*
 * scds_svc_wait is a subroutine in a START method to
 * check that the service is fully available before returning.
 * Calls svc_probe() to check service availability.
 */
int
svc_wait(scds_handle_t handle)
{
    while (1) {
        /* Wait for 5 seconds */
        if (scds_svc_wait(handle, 5) != SCHA_ERR_NOERR) {
            scds_syslog(LOG_ERR, "Service failed to start.");
            return (1);        /* Start Failure */
        }
        /* Check if service is fully up every 5 seconds */
        if (svc_probe(handle) == 0) {
            scds_syslog(LOG_INFO, "Service started successfully.");
            return (0);
        }
    }
    return (0);
}
```

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scds_initialize(3HA), scds_pmf_start(3HA) , scha_calls(3HA), `attributes`(5), r_properties(5)

Notes

- If the START method exceeds the `Start_timeout` setting on the resource, the RGM will kill the START method even if the START method is currently waiting for `scds_svc_wait()` to return.
- If `Retry_interval` on the resource is larger then `Start_timeout`, the START method could be timed out by the RGM even if the number of failures is below `Retry_count`.

- If a START method starts multiple process groups with multiple calls to `scds_pmf_start()`, `scds_svc_wait()` starts process groups as they die. It does not enforce any dependencies between process groups. Do not use `scds_svc_wait()` if there is a dependency between process groups such that failure of one process group requires a restart of other process groups. Instead, use `sleep()` to wait between health checks of the process groups.

Name scds_syslog – write a message to the system log

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void scds_syslog(int priority, const char*format...
```

Description The `scds_syslog()` function writes a message to the system log. It uses the facility returned by thescha_cluster_getlogfacility(3HA) function. You can forward these messages to appropriate log files and users. See `syslog.conf`(4) for more information.

All syslog messages are prefixed with:
`SC[`<*resourceTypeName*>`,`<*resourceGroupName*>`,`<*resourceName*>`,`<*methodName*>

Caution – Messages written to the system log are not internationalized. Do not use `gettext()` or other message translation functions in conjunction with this function.

Parameters The following parameters are supported:

priority	Message priority, as specified by `syslog`(3C)
format	Message format string, as specified by `printf`(3C)
...	Variables, indicated by the *format* parameter, as specified by `printf()`

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also `printf`(3C), scds_syslog_debug(3HA), scha_cluster_getlogfacility(3HA), `syslog`(3C), `syslog.conf`(4), `attributes`(5)

Name scds_syslog_debug – write a debugging message to the system log

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

void  scds_syslog_debug(int debug_level, constchar *format...
```

Description The `scds_syslog_debug()` function writes a debugging message to the system log. It uses the facility returned by the scha_cluster_getlogfacility(3HA) function.

All syslog messages are prefixed with:
`SC[`*<resourceTypeName>*`,`*<resourceGroupName>*`,`*<resourceName>*`,`*<methodName>*

If you specify a *debug_level* greater than the current debugging level being used, no information is written.

The DSDL defines the maximum debugging level, `SCDS_MAX_DEBUG_LEVEL`, as 9. The scds_initialize(3HA) function, which the calling program must call before `scds_syslog_debug()`, retrieves the current debugging level from the file: `/var/cluster/rgm/rt/`*<resourceTypeName>*`/loglevel`.

Caution – Messages written to the system log are not internationalized. Do not use `gettext()` or other message translation functions in conjunction with this function.

Parameters The following parameters are supported:

debug_level	Debugging level at which this message is to be written. Valid debugging levels are between 1 and `SCDS_MAX_DEBUG_LEVEL`, which is defined as 9 by the DSDL. If the specified debugging level is greater than the debugging level set by the calling program, the message is not written to the system log.
format	Message format string, as specified by `printf`(3C)
...	Variables, indicated by the *format* parameter, as specified by `printf`(3C)

Examples

EXAMPLE 1 Display All Debugging Messages

To see all debugging messages for resource type `SUNW.iws`, issue the following command on all nodes of your cluster

```
echo 9 > /var/cluster/rgm/rt/SUNW.iws/loglevel
```

EXAMPLE 2 Suppress Debugging Messages

To suppress debugging messages for resource type `SUNW.iws`, issue the following command on all nodes of your cluster

```
echo 0 > /var/cluster/rgm/rt/SUNW.iws/loglevel
```

Files

`/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also `printf`(3C), scds_syslog(3HA), scha_cluster_getlogfacility(3HA), `syslog`(3C), `syslog.conf`(4), `attributes`(5)

Name scds_timerun – execute a given command in a given amount of time

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  dsdev
#include <rgm/libdsdev.h>

scha_err_t scds_timerun(scds_handle_t handle, const char *command,
     time_t timeout, int signal, int *cmd_exit_code
```

Description The `scds_timerun()` function executes a specified command using hatimerun(1M). If the command does not complete within the allotted time period,which is specified by the *timeout* argument, `scds_timerun()` sends a signal, specified by the *signal* argument, to kill it.

The *command* argument does not support I/O redirection. However, you can write a script to perform redirection and then identify this script in the *command* argument as the command for `scds_timerun()` to execute.

Parameters The following parameters are supported:

handle	The handle returned from scds_initialize(3HA)
command	String containing the command to run
timeout	Time, in seconds, allotted to run the command
signal	Signal to kill the command if it is still running when the timeout expires. If `signal = -1`, then `SIGKILL` is used. See `signal`(3HEAD).
cmd_exit_code	Return code from execution of the command

Return Values The `scds_timerun()` function returns the following:

`0`	The function succeeded.
`non-zero`	The function failed.

Errors

`SCHA_ERR_NOERR`	The command executed and `cmd_exit_code` contains the child program's exit status.
`SCHA_ERR_INTERNAL`	The timeout did not occur, but some other error was detected by `scds_timerun()` that was not an error detected by the child program. Or hatimerun(1M) caught the signal `SIGTERM`.
`SCHA_ERR_INVAL`	There was an invalid input argument.
`SCHA_ERR_TIMEOUT`	The timeout occurred before the command specified by the *command* argument finished executing.

See scha_calls(3HA) for a description of other error codes.

Files `/usr/cluster/include/rgm/libdsdev.h`
Include file

`/usr/cluster/lib/libdsdev.so`
Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also hatimerun(1M), scds_initialize(3HA), scha_calls(3HA), `signal`(3HEAD), `attributes`(5)

Name scha_calls – Sun Cluster library functions used in the implementation of callback methods and monitors of resource types

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

scha_err_t scha_get_function(handle, const char *tag...);

scha_err_t scha_control(const char *tag...
```

Description The Sun Cluster library functions scha_resource_get(3HA), scha_resourcetype_get(3HA), scha_resourcegroup_get(3HA), scha_cluster_get(3HA), scha_control(3HA), scha_strerror(3HA), and scha_resource_setstatus(3HA) provide an interface to be used in the implementation of callback methods and monitors of resource types. The resource types represent services that are controlled by the cluster's Resource Group Manager (RGM) facility.

The "get" functions access cluster configuration information. All these functions have the same general signature. These functions take a *handle* argument that is returned from a previous call to an "open" function. This *handle* indicates the object in the cluster configuration that is to be accessed. A *tag* argument indicates the property of the object that is to be accessed. The value of *tag* determines whether additional arguments are needed and the type of a final "out" argument through which the requested information is returned. You can make repeated "get" calls with the same handle until a "close" call, which invalidates the handle and frees memory that is allocated for values that are returned from the "get" calls.

Memory, if needed to return a value, is allocated for each "get" call. Space allocated to return a value in one call will not be overwritten and reused by subsequent calls.

The scha_control(3HA) function also has a *tag* argument that indicates a control operation, but does not return information in an output argument.

The scha_resource_setstatus(1HA) command sets the `Status` and `Status_msg` properties of a resource that is managed by the RGM.

The man pages for the individual functions should be referred to for the macro values accepted as *tag* argument values for each function, and variable argument types for each *tag*. The types of output arguments are described in the next section.

There is one set of `scha_err_t` `enum`-type return values for the `scha` functions. The enum symbols, integer values, and meaning of the exit codes are described in RETURN VALUES.

The scha_strerror(3HA) function converts an `scha_err_t` code returned by an `scha` function to the appropriate error message.

Output Argument Data Types

`uint_t`
An unsigned integer type. This type is defined in the system header file <`sys/types.h`>.

`boolean_t`
This type is defined in the system header file <`sys/types.h`>.

```
typedef enum { B_FALSE, B_TRUE } boolean_t;
```

`scha_switch_t`
An `enum` type that indicates an `On_Off_switch` or `Monitored_switch` resource property value.

```
typedef enum scha_switch {
    SCHA_SWITCH_DISABLED = 0,
    SCHA_SWITCH_ENABLED
} scha_switch_t;
```

`scha_rsstate_t`
An `enum` type that indicates a resource state.

```
typedef enum scha_rsstate {
    SCHA_RSSTATE_ONLINE = 0,
    SCHA_RSSTATE_OFFLINE,
    SCHA_RSSTATE_START_FAILED,
    SCHA_RSSTATE_STOP_FAILED,
    SCHA_RSSTATE_MONITOR_FAILED,
    SCHA_RSSTATE_ONLINE_NOT_MONITORED,
    SCHA_RSSTATE_STARTING,
    SCHA_RSSTATE_STOPPING
} scha_rsstate_t;
```

`scha_rgstate_t`
An `enum` type that indicates a resource group state.

```
typedef enum scha_rgstate {
    SCHA_RGSTATE_UNMANAGED = 0,
    SCHA_RGSTATE_ONLINE,
    SCHA_RGSTATE_OFFLINE,
    SCHA_RGSTATE_PENDING_ONLINE,
    SCHA_RGSTATE_PENDING_OFFLINE,
    SCHA_RGSTATE_ERROR_STOP_FAILED
    SCHA_RGSTATE_ONLINE_FAULTED,
    SCHA_RGSTATE_PENDING_ONLINE_BLOCKED
} scha_rgstate_t;
```

`scha_rgmode_t`
An `enum` type that indicates if the mode of a resource group is failover or scalable.

```
typedef enum scha_rgmode {
    RGMODE_NONE = 0,
    RGMODE_FAILOVER,
    RGMODE_SCALABLE
} scha_rgmode_t;
```

`scha_failover_mode_t`
An `enum` type that indicates a value for the `Failover_Mode` resource property.

```
typedef enum scha_failover_mode {
    SCHA_FOMODE_NONE = 0,
    SCHA_FOMODE_HARD,
    SCHA_FOMODE_SOFT,
    SCHA_FOMODE_RESTART_ONLY,
    SCHA_FOMODE_LOG_ONLY
} scha_failover_mode_t;
```

`scha_initnodes_flag_t`
An `enum` type that indicates a value for the `Init_nodes` resource type property.

```
typedef enum scha_initnodes_flag {
    SCHA_INFLAG_RG_PRIMARIES = 0,
    SCHA_INFLAG_RT_INSTALLED_NODES
} scha_initnodes_flag_t;
```

`scha_node_state_t`
An `enum` type that indicates whether a node is up or down.

```
typedef enum scha_node_state {
    SCHA_NODE_UP = 0,
    SCHA_NODE_DOWN
} scha_node_state_t;
```

`scha_str_array_t`
A structure that holds the value of a list of strings.

```
typedef struct scha_str_array {
    uint_t       array_cnt;
    boolean_t    is_ALL_value;
    char         **str_array;
} scha_str_array_t;
```

`array_cnt`	Gives the number elements in the list.
`is_ALL_value`	If a property is set to the "all" value, also known as the wild card or asterisk (*) character, `is_ALL_value` is set to `B_TRUE` and `str_array` is `NULL`. As a result, `str_array` is ignored.
`str_array`	A pointer to an array of `array_cnt` `strings`.

`scha_uint_array_t`
A structure that holds the value of a list of unsigned integers.

```
typedef struct scha_uint_array {
    uint_t    array_cnt;
    uint_t    *int_array;
} scha_uint_array_t;
```

`array_cnt`	The number of elements in the list.
`int_array`	A pointer to an array of `array_cnt` unsigned integers.

`scha_status_value_t`
The structure for returning the status and status message of a resource.

```
typedef struct scha_status_value {
    scha_rsstatus_t        status;
    char                   *status_msg;
} scha_status_value_t;

typedef enum scha_rsstatus {
    SCHA_RSSTATUS_ONLINE = 0,
    SCHA_RSSTATUS_OFFLINE,
    SCHA_RSSTATUS_FAULTED,
    SCHA_RSSTATUS_DEGRADED,
    SCHA_RSSTATUS_UNKNOWN
} scha_rsstatus_t;
```

`status` — Holds an `enum` value that indicates the resource status as set by the resource monitor.

`scha_extprop_value_t`
The structure that is used for returning the value of an extension property.

The `prop_type` structure member indicates the type of the extension property and determines which element of the union is used for the `prop_type` field and the return values:

```
SCHA_PTYPE_STRING        val_str
SCHA_PTYPE_INT           val_int
SCHA_PTYPE_ENUM          val_enum
SCHA_PTYPE_BOOLEAN       val_boolean
SCHA_PTYPE_STRINGARRAY   val_strarray


typedef struct scha_extprop_value {
    scha_prop_type_t        prop_type;
    union {
          char              *val_str;
          int               val_int;
          char              *val_enum;
          boolean_t         val_boolean;
          scha_str_array_t  *val_strarray;
          } val;
} scha_extprop_value_t;
```

Return Values The following is a list of the `scha_err_t` error numbers and the error codes returned by scha_strerror(3HA).

`0 SCHA_ERR_NOERR` — No error was found.

`1 SCHA_ERR_NOMEM` — Not enough swap.

2 SCHA_ERR_HANDLE	Invalid resource management handle.
3 SCHA_ERR_INVAL	Invalid input argument.
4 SCHA_ERR_TAG	Invalid API tag.
5 SCHA_ERR_RECONF	Cluster is reconfiguring.
6 SCHA_ERR_ACCESS	Permission denied.
7 SCHA_ERR_SEQID	Resource, resource group, or resource type has been updated since last scha_*_open call.
8 SCHA_ERR_DEPEND	Object dependency problem.
9 SCHA_ERR_STATE	Object is in wrong state.
10 SCHA_ERR_METHOD	Invalid method.
11 SCHA_ERR_NODE	Invalid node.
12 SCHA_ERR_RG	Invalid resource group.
13 SCHA_ERR_RT	Invalid resource type.
14 SCHA_ERR_RSRC	Invalid resource.
15 SCHA_ERR_PROP	Invalid property.
16 SCHA_ERR_CHECKS	Sanity checks failed.
17 SCHA_ERR_RSTATUS	Bad resource status.
18 SCHA_ERR_INTERNAL	Internal error was encountered.
31 SCHA_ERR_TIMEOUT	Operation timed out.
32 SCHA_ERR_FAIL	Failover attempt failed.

Files

/usr/cluster/include/scha.h	Include file
/usr/cluster/lib/libscha.so	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scha_cmds(1HA), scha_resource_setstatus(1HA)scha_cluster_get(3HA), scha_control(3HA), scha_resource_get(3HA), scha_resourcegroup_get(3HA), scha_resource_setstatus(3HA), scha_resourcetype_get(3HA), scha_strerror(3HA), `attributes`(5)

Name scha_cluster_getlogfacility – cluster log facility access

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

 scha_err_t scha_cluster_getlogfacility(int *logfacility);
```

Description The `scha_cluster_getlogfacility()` function returns the system log facility number that is being used as the cluster log. The value is intended to be used with the Solaris `syslog`(3C) function by resource type implementations to record events and status messages to the cluster log.

The function returns an error status, and if successful, the facility number in the location pointed to by the *logfacility* argument.

Return Values The `scha_cluster_getlogfacility()` function returns the following:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	The function succeeded.

See scha_calls(3HA) for a description of other error codes.

Examples **EXAMPLE 1** Using the `scha_cluster_getlogfacility()` Function

```
main()
{
    scha_err_t    err_code;
    int logfacility;

    err_code = scha_cluster_getlogfacility(&logfacility);

    if (err_code == SCHA_ERR_NOERR) {
        openlog("test resource", LOG_CONS, logfacility);
        syslog(LOG_INFO, "Access function call succeeded.");
    }
}
```

Files

`/usr/cluster/include/scha.h`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also `syslog`(3C), scha_calls(3HA), scha_cluster_get(3HA), scha_strerror(3HA), `attributes`(5)

Name scha_cluster_getnodename – local cluster node name access function

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

 scha_err_t scha_cluster_getnodename(char **nodename);
```

Description The `scha_cluster_getnodename()` function returns the name of the cluster node on which the function is called. The node name is not necessarily the same as the Solaris system name. The function returns an error status, and if successful, a string that contains the node name in the location that is pointed to by the *nodename* argument.

If the call fails, the *nodename* is set to `NULL`. The caller of `scha_cluster_getnodename()` is responsible for freeing the memory that is allocated for the returned string by using the standard C library function `free`(3C). Freeing the memory is required only if the function succeeds.

Return Values The `scha_cluster_getnodename()` function returns the following values:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Function succeeded.

See scha_calls(3HA) for a description of other error codes.

Examples **EXAMPLE 1** Using the `scha_cluster_getnodename()` Function

```
scha_err_t  err_code;
char *nodename;
err_code = scha_cluster_getnodename(&nodename);
...
if (nodename != NULL) free(nodename);
```

Files

`/usr/cluster/include/scha.h`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also `free`(3C), scha_calls(3HA), scha_cluster_get(3HA), scha_cluster_getzone(3HA), scha_strerror(3HA), `attributes`(5)

Name scha_cluster_getzone – zone name access function

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

 scha_err_t scha_cluster_getzone(char **zonename);
```

Description The `scha_cluster_getzone()` function returns a string that identifies the zone from which the function is called. If you call this function in a non-global zone, the zone name and the node name are returned, in the format `nodename:zonename`. If you call this function in the global zone, only the node name is returned. The node name is not necessarily the same as the Solaris system name. The function returns an error status. If successful, the function also returns a string that contains the node name and the zone name in the location that is pointed to by the *zonename* argument.

If the call fails, the *zonename* argument is set to `NULL`. The caller of `scha_cluster_getzone()` is responsible for freeing the memory that is allocated for the returned string by using the standard C library function `free`(3C). Freeing the memory is required only if the function succeeds.

Return Values The `scha_cluster_getzone()` function returns the following values:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	The function succeeded.

See scha_calls(3HA) for a description of other error codes.

Examples **EXAMPLE 1** Using the `scha_cluster_getzone()` Function

```
scha_err_t  err_code;
char *zonename;
err_code = scha_cluster_getzone(&zonename);
...
if (zonename != NULL) free(zonename);
```

Files

`/usr/cluster/include/scha.h`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also `free`(3C), scha_calls(3HA), scha_cluster_get(3HA), scha_cluster_getnodename(3HA), scha_strerror(3HA), `attributes`(5)

Name scha_cluster_open, scha_cluster_close, scha_cluster_get – cluster information access functions

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

scha_err_t scha_cluster_open(scha_cluster_t *handle);

scha_err_t scha_cluster_get(scha_cluster_t handle, const char **tag, ...);

scha_err_t scha_cluster_close(scha_cluster_t handle);
```

Description The `scha_cluster_open()`, `scha_cluster_get()`, and `scha_cluster_close()` functions are used together to obtain information about a cluster.

`scha_cluster_open()` initializes cluster access and returns an access handle to be used by `scha_cluster_get()`. The *handle* argument is the address of a variable to hold the value that is returned by the function call.

`scha_cluster_get()` accesses cluster information as indicated by the *tag* argument. If you call this function in a non-global zone, the zone name and the node name are returned. The *handle* argument is a value that is returned from a prior call to `scha_cluster_open()`. The *tag* argument is a string value that is defined by a macro in the <`scha_tags.h`> header file. The arguments that follow the tag depend on the value of the *tag* argument.

You might need to provide an additional argument after the *tag* argument to indicate a cluster node or zone from which the information is to be retrieved. The last argument in the argument list is to be of a type that is suitable to hold the information that is indicated by the *tag* argument. This argument is the output argument for the cluster information. No value is returned for the output argument if the function fails. Memory that is allocated to hold information that is returned by the `scha_cluster_get()` function remains intact until `scha_cluster_close()` is called on the handle that is used for the `scha_cluster_get()` function.

`scha_cluster_close()` takes a `handle` argument that is returned from a previous call to the `scha_cluster_get()` function. This function invalidates the handle and frees memory that is allocated to return values to `scha_cluster_get()` calls that were made with the handle. Note that memory, if needed to return a value, is allocated for each `get` call. Space allocated to return a value in one call is not overwritten and reused by subsequent calls.

Macros That You Can Use for *tag* Arguments

Macros that are defined in <`scha_tags.h`> that you can use as *tag* arguments follow. The type of the output argument and any additional arguments are indicated. Structure and `enum` types are described in scha_calls(3HA).

`SCHA_ALL_NODEIDS`
The output argument type is `scha_uint_array_t**`.

This macro returns numeric node identifiers of all the nodes in the cluster.

`SCHA_ALL_NODENAMES`
The output argument type is `scha_str_array_t**`.

This macro returns the names of all nodes in the cluster.

`SCHA_ALL_NONGLOBAL_ZONES`
The output argument type is `scha_str_array_t**`.

This macro returns zone names of all the non-global zones on all nodes in the cluster. The format of the zone name that is returned is `nodename:zonename`.

`SCHA_ALL_NONGLOBAL_ZONES_NODEID`
The output argument type is `scha_str_array_t**`.

This macro returns zone names of all the non-global zones on the cluster node whose numeric node identifier is given as the argument. The format of the zone name that is returned is `nodename:zonename`.

`SCHA_ALL_PRIVATELINK_HOSTNAMES`
The output argument type is `scha_str_array_t**`.

This macro returns the host names for all cluster nodes or zones by which the nodes or zones are addressed on the cluster interconnect.

`SCHA_ALL_RESOURCEGROUPS`
The output argument type is `scha_str_array_t**`.

This macro returns the names of all the resource groups that are being managed on the cluster.

`SCHA_ALL_RESOURCETYPES`
The output argument type is `scha_str_array_t**`.

This macro returns the names of all the resource types that are registered on the cluster.

`SCHA_ALL_ZONES`
The output argument type is `scha_str_array_t**`.

This macro returns zone names of all zones, including the global zone, on all nodes in the cluster. The format of the zone name that is returned is `nodename:zonename`.

`SCHA_ALL_ZONES_NODEID`
The output argument type is `scha_str_array_t**`.

This macro returns zone names of all zones, including the global zone, on the cluster node whose numeric node identifier is given as the argument. The format of the zone name that is returned is `nodename:zonename`.

`SCHA_CLUSTERNAME`
The output argument is type `char**`.

This macro returns the name of the cluster.

`SCHA_NODEID_LOCAL`
The output argument type is `uint_t*`.

This macro returns the numeric node identifier for the node where the command is executed.

`SCHA_NODEID_NODENAME`
The output argument type is `uint_t*`. An additional argument is of type `char *`. The macro requires an additional argument that is a name of a cluster node.

This macro returns the numeric node identifier of the node indicated by the name.

`SCHA_NODENAME_LOCAL`
The output argument type is `char**`.

This macro returns the name of the cluster node where the function is executed.

`SCHA_NODENAME_NODEID`
The output argument type is `char**`. An additional argument is of type `uint_t`. The additional argument is a numeric cluster node identifier.

This macro returns the name of the node indicated by the numeric identifier.

`SCHA_NODESTATE_LOCAL`
The output argument type is `scha_node_state_t*`.

This macro returns `SCHA_NODE_UP` or `SCHA_NODE_DOWN`, depending on the state of the node where the command is executed.

`SCHA_NODESTATE_NODE`
The output argument type is `scha_node_state_t*`. An additional argument is type `char*`. The macro requires an additional unflagged argument that is the name of a cluster node.

This macro returns `SCHA_NODE_UP` or `SCHA_NODE_DOWN`, depending on the state of the named node.

`SCHA_PRIVATELINK_HOSTNAME_LOCAL`
The output argument type is `char**`.

This macro returns the host name by which the node or zone on which the command is run is addressed on the cluster interconnect.

`SCHA_PRIVATELINK_HOSTNAME_NODE`
The output argument type is `char**`. An additional argument is of type `char *`. This macro requires an additional unflagged argument that is the name of a cluster node.

This macro returns the host name by which the named node or zone is addressed on the cluster interconnect.

`SCHA_SYSLOG_FACILITY`
The output argument type is `int*`.

This macro returns the number of the `syslog`(3C) facility that the RGM uses for log messages. The value that is returned is `24`, which corresponds to the `LOG_DAEMON` facility value.

`SCHA_ZONE_LOCAL`
The output argument type is `scha_str_array_t**`.

This macro returns the name of the zone from which the call is made. The format of the zone name that is returned is `nodename:zonename`.

Errors

`SCHA_ERR_NOERR`	The function succeeded.

See scha_calls(3HA) for a description of other error codes.

Examples

EXAMPLE 1 Using the `scha_cluster_get()` Function

The following example uses the `scha_cluster_get()` function to get the names of all cluster nodes. The function also determines whether the node is up or down.

```
#include <scha.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
        scha_err_t              err;
        scha_node_state_t       node_state;
        scha_str_array_t        *all_nodenames;
        scha_cluster_t          handle;
        int                     ix;
        const char              *str;

        err = scha_cluster_open(&handle);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_open()0);
                exit(err);
        }

        err = scha_cluster_get(handle, SCHA_ALL_NODENAMES, &all_nodenames);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_get()0);
                exit(err);
        }

        for (ix = 0; ix < all_nodenames->array_cnt; ix++) {
                err = scha_cluster_get(handle, SCHA_NODESTATE_NODE,
                    all_nodenames->str_array[ix], &node_state);
                if (err != SCHA_ERR_NOERR) {
                        fprintf(stderr, "FAILED: scha_cluster_get()"
```

EXAMPLE 1 Using the `scha_cluster_get()` Function *(Continued)*

```
                            "SCHA_NODESTATE_NODE0);
                        exit(err);
                }

                switch (node_state) {
                case SCHA_NODE_UP:
                        str = "UP";
                        break;
                case SCHA_NODE_DOWN:
                        str = "DOWN";
                        break;
                }

                printf("State of node: %s value: %s\
",
                    all_nodenames->str_array[ix], str);
        }
}
```

Files

`/usr/cluster/include/scha.h`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Evolving

See Also scha_cluster_get(1HA), scha_calls(3HA), scha_cluster_getlogfacility(3HA), scha_cluster_getnodename(3HA), scha_strerror(3HA), `syslog`(3C), `attributes`(5)

Name scha_control, scha_control_zone – resource and resource group control request functions

Synopsis

```
cc [flags...] -I/usr/cluster/include file -L/usr/cluster/lib  -l  scha
#include <scha.h>

 scha_err_t scha_control(const char *tag, const char *rgname, const char *rname);

 scha_err_t scha_control_zone(const char *tag, const char *rgname,
     const char *rname, const char *zonename);
```

Description The `scha_control()` and `scha_control_zone()` functions each provide an interface to request the restart or relocation of a resource or a resource group that is under the control of the Resource Group Manager (RGM). Use these functions in resource monitors.

Use the `scha_control()` and `scha_control_zone()` functions only for resource types whose `Global_zone` property is set to `TRUE`. These functions are not needed if the `Global_zone` property is set to `FALSE`. For more information, see the rt_properties(5) man page.

The setting of the `Failover_mode` property of the indicated resource might suppress the requested `scha_control()` or `scha_control_zone()` action. If `Failover_mode` is `RESTART_ONLY`, only `SCHA_RESOURCE_RESTART` is permitted. Other requests, including `SCHA_GIVEOVER`, `SCHA_CHECK_GIVEOVER`, `SCHA_RESTART`, and `SCHA_CHECK_RESTART`, return the `SCHA_ERR_CHECKS` exit code and the requested giveover or restart action is not executed, producing only a `syslog` message. If the `Retry_count` and `Retry_interval` properties are set on the resource, the number of resource restarts is limited to `Retry_count` attempts within the `Retry_interval`. If `Failover_mode` is `LOG_ONLY`, any `scha_control()` or `scha_control_zone()` giveover, restart, or disable request returns the `SCHA_ERR_CHECKS` exit code and the requested giveover or restart action is not executed, producing only a `syslog` message.

tag Arguments The *tag* argument indicates whether the request is to restart or relocate the resource or resource group. This argument should be a string value that is defined by one of the following macros, which are defined in `<scha_tags.h>`:

`SCHA_CHANGE_STATE_OFFLINE`
: Requests that the proxy resource that is named by the `rname` argument be brought offline on the local node or zone. A *proxy resource* is a Sun Cluster resource that imports the state of a resource from another cluster such as Oracle Cluster Ready Services (CRS). Oracle CRS, which is now known as Oracle clusterware CRS, is a platform-independent set of system services for cluster environments. This change in state reflects, in the context of the Sun Cluster software, the change in state of the external resource.

  When you change the state of a proxy resource with this *tag* argument, methods of the proxy resource are not executed.

  If a fault occurs on a "depended-on" resource on a node or zone, and the resource cannot recover, the monitor brings that resource on that node or zone offline. The monitor brings the resource offline by calling the `scha_control()` or `scha_control_zone()` function with

the SCHA_RESOURCE_DISABLE request. The monitor also brings all of the depended-on resource's offline-restart dependents offline by triggering a restart on them. When the cluster administrator resolves the fault and reenables the depended-on resource, the monitor brings the depended-on resource's offline-restart dependents back online as well.

SCHA_CHANGE_STATE_ONLINE
Requests that the proxy resource that is named by the rname argument be brought online on the local node or zone. A *proxy resource* is a Sun Cluster resource that imports the state of a resource from another cluster such as CRS. This change in state reflects, in the context of the Sun Cluster software, the change in state of the external resource.

When you change the state of a proxy resource with this *tag* argument, methods of the proxy resource are not executed.

SCHA_CHECK_GIVEOVER
Performs all the same validity checks that would be done for a SCHA_GIVEOVER of the resource group named by the *rgname* argument, but does not actually relocate the resource group.

SCHA_CHECK_RESTART
Performs all the same validity checks that would be done for a SCHA_RESTART request of the resource group named by the *rgname* argument, but does not actually restart the resource group.

The SCHA_CHECK_GIVEOVER and SCHA_CHECK_RESTART requests are intended to be used by resource monitors that take direct action upon resources, for example, killing and restarting processes, rather than invoking the scha_control() or scha_control_zone() function to perform a giveover or restart. If the check fails, the monitor should sleep and restart its probes rather than invoke its failover actions. See ERRORS.

The *rgname* argument is the name of the resource group that is to be restarted or relocated. If the group is not online on the node or zone where the request is made, the request is rejected.

The *rname* argument is the name of a resource in the resource group. Presumably this is the resource whose monitor is making the scha_control() or scha_control_zone() request. If the named resource is not in the resource group, the request is rejected.

The exit code of the command indicates whether the requested action was rejected. If the request is accepted, the function does not return until the resource group or resource has completed going offline and back online. The fault monitor that called the scha_control() or scha_control_zone() function might be stopped as a result of the resource group's going offline and so might never receive the return status of a successful request.

SCHA_GIVEOVER
Requests that the resource group named by the *rgname* argument be brought offline on the local node or zone, and online again on a different node or zone of the RGM's choosing. Note that, if the resource group is currently online on two or more nodes or zones and

there are no additional available nodes or zones on which to bring the resource group online, it can be taken offline on the local node or zone without being brought online elsewhere. The request might be rejected depending on the result of various checks. For example, a node or zone might be rejected as a host because the group was brought offline due to a `SCHA_GIVEOVER` request on that node or zone within the interval specified by the `Pingpong_interval` property.

If the cluster administrator configures the `RG_affinities` properties of one or more resource groups and if you issue a **`scha_control GIVEOVER`** request on one resource group, more than one resource group might be relocated. The `RG_affinities` property is described in rg_properties(5).

The `MONITOR_CHECK` method is called before the resource group that contains the resource is relocated to a new node or zone as the result of a call to the `scha_control()` or `scha_control_zone()` function or the issuing of the `scha_control` or `scha_control_zone` command from a fault monitor. See the scha_control(1HA) man page.

The `MONITOR_CHECK` method may be called on any node or zone that is a potential new master for the resource group. The `MONITOR_CHECK` method is intended to assess whether a node or zone is running well enough to run a resource. The `MONITOR_CHECK` method must be implemented in such a way that it does not conflict with the running of another method concurrently.

Failure of the `MONITOR_CHECK` method vetoes the relocation of the resource group to the node or zone where the callback was invoked.

`SCHA_IGNORE_FAILED_START`

Requests that failure of the currently executing `Prenet_start` or `Start` method should not cause a failover of the resource group, despite the setting of the `Failover_mode` property.

In other words, this request overrides the recovery action that is normally taken for a resource for which the `Failover_Mode` property is set to `SOFT` or `HARD` when that resource fails to start. Normally, the resource group fails over to a different node or zone. Instead, the resource behaves as if `Failover_Mode` is set to `NONE`. The resource enters the `START_FAILED` state, and the resource group ends up in the `ONLINE_FAULTED` state, if no other errors occur.

This request is meaningful only when it is called from a `Start` or `Prenet_start` method that subsequently exits with a nonzero status or times out. This request is valid only for the current invocation of the `Start` or `Prenet_start` method. The `scha_control()` or `scha_control_zone()` function should be called with this request in a situation in which the `Start` method has determined that the resource cannot start successfully on another node or zone. If this request is called by any other method, the error `SCHA_ERR_INVAL` is returned. This request prevents the “ping pong” failover of the resource group that would otherwise occur. See the scha_calls(3HA) man page for a description of the `SCHA_ERR_INVAL` error code.

SCHA_RESOURCE_DISABLE
: Disables the resource that is named by the rname argument on the node or zone on which the scha_control() or scha_control_zone() function is called.

  If a fault occurs on a "depended-on" resource on a node or zone and if the resource cannot recover, the monitor brings that resource on that node or zone offline. The monitor brings the resource offline by calling the scha_control() or scha_control_zone() function with the SCHA_RESOURCE_DISABLE request. The monitor also brings all of the depended-on resource's offline-restart dependents offline by triggering a restart on them. When the cluster administrator resolves the fault and reenables the depended-on resource, the monitor brings the depended-on resource's offline-restart dependents back online as well.

SCHA_RESOURCE_IS_RESTARTED
: Requests that the resource restart counter for the resource named by the rname argument be incremented on the local node or zone, without actually restarting the resource.

  A resource monitor that restarts a resource directly without calling the scha_control() or scha_control_zone() function with the SCHA_RESOURCE_RESTART request (for example, using the pmfadm(1M) command) can use this request to notify the RGM that the resource has been restarted. This fact is reflected in subsequent calls to the scha_resource_get() function with NUM_RESOURCE_RESTARTS queries.

  If the resource's type fails to declare the Retry_interval standard property, the SCHA_RESOURCE_IS_RESTARTED request of the scha_control() or scha_control_zone() function is not permitted and the scha_control() or scha_control_zone() function returns error code 13 (SCHA_ERR_RT).

SCHA_RESOURCE_RESTART
: Requests that the resource named by the rname argument be brought offline and online again on the local node or zone, without stopping any other resources in the resource group. The resource is stopped and started by applying the following sequence of methods to it on the local node or zone:

  ```
  MONITOR_STOP
  STOP
  START
  MONITOR_START
  ```

  If the resource's type does not declare a MONITOR_STOP and MONITOR_START method, only the STOP and START methods are invoked to perform the restart. The resource's type must declare a START and STOP method. If the resource's type does not declare both a START and STOP method, the scha_control() or scha_control_zone() function fails with error code 13 (SCHA_ERR_RT). See the scha_calls(3HA) man page for a description of the SCHA_ERR_RT error code.

If a method invocation fails while restarting the resource, the RGM might set an error state, or relocate the resource group, or reboot the node or zone, depending on the setting of the `Failover_mode` property of the resource. For additional information, see the `Failover_mode` property in r_properties(5).

A resource monitor using this request to restart a resource can use the `NUM_RESOURCE_RESTARTS` query of `scha_resource_get()` to keep count of recent restart attempts.

Resource types that have `PRENET_START` or `POSTNET_STOP` methods need to use the `SCHA_RESOURCE_RESTART` request with care. Only the `MONITOR_STOP`, `STOP`, `START`, and `MONITOR_START` methods are applied to the resource. Network address resources on which this resource depends are not restarted and remain online.

If a fault occurs on a "depended-on" resource on a node or zone, and the resource cannot recover, the monitor brings that resource on that node or zone offline. The monitor brings the resource offline by calling the `scha_control()` or `scha_control_zone()` function with the `SCHA_RESOURCE_DISABLE` request. The monitor also brings all of the depended-on resource's offline-restart dependents offline by triggering a restart on them. When the cluster administrator resolves the fault and reenables the depended-on resource, the monitor brings the depended-on resource's offline-restart dependents back online as well.

`SCHA_RESTART`
Requests that the resource group named by the *rgname* argument be brought offline, then online again, without forcing relocation to a different node or zone. The request may ultimately result in relocating the resource group if a resource in the group fails to restart. A resource monitor using this request to restart a resource group can use the `NUM_RG_RESTARTS` query of `scha_resource_get()` to keep count of recent restart attempts.

Return Values These functions return the following values:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	The function succeeded.
`SCHA_ERR_CHECKS`	The request was rejected. The checks on relocation failed.

See the scha_calls(3HA) man page for a description of other error codes.

Normally, a fault monitor that receives an error code from the `scha_control()` or the `scha_control_zone()` function should sleep for awhile and then restart its probes. These functions must do so because some error conditions resolve themselves after awhile. An example of such an error condition is the failover of a global device service, which causes disk resources to become temporarily unavailable. After the error condition has resolved, the resource itself might become healthy again. If not, a subsequent `scha_control()` or `scha_control_zone()` request might succeed.

Files

`</usr/cluster/include/scha.h>`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Evolving

See Also rt_callbacks(1HA), scha_control(1HA), pmfadm(1M), scha_calls(3HA), scha_resource_open(3HA), scha_strerror(3HA), `attributes`(5), r_properties(5), rg_properties(5), rt_properties(5)

Name scha_resourcegroup_open, scha_resourcegroup_close, scha_resourcegroup_get – resource information access functions

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

scha_err_t scha_resourcegroup_open(const char *rgname, scha_resourcegroup_t *handle);

scha_err_t scha_resourcegroup_close(scha_resourcegroup_t handle);

scha_err_t scha_resourcegroup_get(scha_resourcegroup_t handle, const char *tag...);
```

Description The `scha_resourcegroup_open()`, `scha_resourcegroup_get()`, and `scha_resourcegroup_close()` functions are used together to access information about a resource group that is managed by the Resource Group Manager (RGM) cluster facility.

`scha_resourcegroup_open()` initializes access of the resource group and returns a handle to be used by `scha_resourcegroup_get()`.

The *rgname* argument names the resource group to be accessed.

The *handle* argument is the address of a variable to hold the value that is returned by the function.

`scha_resourcegroup_get()` accesses resource group information as indicated by the *tag* argument. The *tag* should be a string value that is defined by a macro in the `<scha_tags.h>` header file. Arguments following the tag depend on the value of *tag*. An additional argument following the tag might be needed to indicate a cluster node from which the information is to be retrieved.

The last argument in the argument list is to be of a type suitable to hold the information indicated by *tag*. This parameter is the output argument for the resource group information that is to be retrieved. No value is returned for the output argument if the function fails. Memory that is allocated to hold information that is returned by `scha_resourcegroup_get()` remains intact until `scha_resourcegroup_close()` is called on the handle that is used for `scha_resourcegroup_get()`.

`scha_resourcegroup_close()` takes a *handle* argument that is returned from a previous call to `scha_resourcegroup_open()`. It invalidates the handle and frees memory that is allocated to return values to `scha_resourcegroup_get()` calls that were made with the handle. Note that memory, if needed to return a value, is allocated for each `get` call. Space that is allocated to return a value in one call is not overwritten or reused by subsequent calls.

Macros That You Can Use for *tag* Arguments You can use the following macros that are defined in `<scha_tags.h>` as *tag* arguments to the `scha_resourcegroup_get()` function. These macros name resource group properties. The value of the property of the resource group is generated. The `RG_STATE` property refers to the value on the node or zone where the function is called.

The type of the output argument and any additional arguments are indicated. Structure and `enum` types are described in the scha_calls(3HA) man page.

`SCHA_DESIRED_PRIMARIES`
The output argument type is `int*`.

`SCHA_FAILBACK`
The output argument type is `boolean_t*`.

`SCHA_GLOBAL_RESOURCES_USED`
The output argument type is `scha_str_array_t**`.

`SCHA_IMPL_NET_DEPEND`
The output argument type is `boolean_t*`.

`SCHA_MAXIMUM_PRIMARIES`
The output argument type is `int*`.

`SCHA_NODELIST`
The output argument type is `scha_str_array_t**`.

`SCHA_PATHPREFIX`
The output argument type is `char**`.

`SCHA_PINGPONG_INTERVAL`
The output argument type is `int*`.

`SCHA_RESOURCE_LIST`
The output argument type is `scha_str_array_t**`.

`SCHA_RG_AFFINITIES`
The output argument type is `char**`.

`SCHA_RG_AUTO_START`
The output argument type is `boolean_t*`.

`SCHA_RG_DEPENDENCIES`
The output argument type is `scha_str_array_t**`.

`SCHA_RG_DESCRIPTION`
The output argument type is `char**`.

`SCHA_RG_IS_FROZEN`
The output argument type is `boolean_t*`.

`SCHA_RG_MODE`
The output argument type is `scha_rgmode_t*`.

`SCHA_RG_PROJECT_NAME`
The output argument type is `char**`.

`SCHA_RG_SLM_CPU`
The output argument type is `char**`.

`SCHA_RG_SLM_CPU_MIN`
The output argument type is `char**`.

`SCHA_RG_SLM_PSET_TYPE`
The output argument type is `char**`.

`SCHA_RG_SLM_TYPE`
The output argument type is `char**`.

`SCHA_RG_STATE`
The output argument type is `scha_rgstate_t*`.

`SCHA_RG_STATE_NODE`
The output argument type is `scha_rgstate_t*`. An additional argument type is `char*`. The additional argument names a cluster node and returns the state of the resource group on that node.

`SCHA_RG_SUSP_AUTO_RECOVERY`
The output argument type is `boolean_t*`.

`SCHA_RG_SYSTEM`
The output argument type is `boolean_t*`.

Return Values These functions return the following:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	Function succeeded.

See scha_calls(3HA) for a description of other error codes.

Examples **EXAMPLE 1** Using the `scha_resourcegroup_get()` Function

The following example uses `scha_resourcegroup_get()` to get the list of resources in the resource group `example_RG`.

```
main() {
   #include <scha.h>

   scha_err_t err;
   scha_str_array_t *resource_list;
   scha_resourcegroup_t handle;
   int ix;

   char * rgname = "example_RG";

   err = scha_resourcegroup_open(rgname, &handle);

   err = scha_resourcegroup_get(handle, SCHA_RESOURCE_LIST, \
         &resource_list);

   if (err == SCHA_ERR_NOERR) {
```

EXAMPLE 1 Using the `scha_resourcegroup_get()` Function *(Continued)*

```
    for (ix = 0; ix < resource_list->array_cnt; ix++) {
        printf("Group: %s contains resource %s\
", rgname,
                resource_list->str_array[ix]);
        }
    }

/* resource_list memory freed */
 err = scha_resourcegroup_close(handle);
}
```

Files

`/usr/cluster/include/scha.h`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scha_resourcegroup_get(1HA) , scha_calls(3HA), `attributes`(5)

Name scha_resource_open, scha_resource_close, scha_resource_get – resource information access functions

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

scha_err_t scha_resource_open(const char *rname, const char *rgname,
     scha_resource_t *handle);

scha_err_t scha_resource_close(scha_resource_t handle);

scha_err_t scha_resource_get(scha_resource_t handle, const char *tag,...);
```

Description The `scha_resource_open()`, `scha_resource_get()`, and `scha_resource_close()` functions are used together to access information about a resource that is managed by the Resource Group Manager (RGM) cluster facility.

`scha_resource_open()` initializes access of the resource and returns a handle to be used by `scha_resource_get()`.

The *rname* argument of `scha_resource_open()` names the resource to be accessed. The *rgname* argument is the name of the resource group in which the resource is configured. The *rgname* argument may be `NULL` if the group name is not known. However, the execution of the function is more efficient if it is provided. The *handle* argument is the address of a variable to hold the value returned from the function call.

`scha_resource_get()` accesses resource information as indicated by the *tag* argument. The *tag* argument should be a string value defined by a macro in the <`scha_tags.h`> header file. Arguments following the tag depend on the value of *tag*. An additional argument following the tag may be needed to indicate a cluster node from which the information is to be retrieved, or other information that is specific to the tag. The last argument in the argument list is to be of a type that is suitable to hold the information that is indicated by *tag*. This argument is the output argument for the resource information. No value is returned for the output argument if the function fails.

Memory that is allocated to hold information returned by `scha_resource_get()` remains intact until `scha_resource_close()` is called on the handle used for the `scha_resource_get()`. Note that repeated calls to `scha_resource_get()` with the same handle and tag cause new memory to be allocated. Space allocated to return a value in one call will not be overwritten and reused by subsequent calls.

`scha_resource_close()` takes a *handle* argument that is returned from a previous call to `scha_resource_open()`. It invalidates the handle and frees memory allocated to return values to `scha_resource_get()` calls that were made with the handle.

Macros defined in <`scha_tags.h`> that may be used as *tag* arguments to `scha_resource_get()` follow.

The type of the output argument and any additional arguments are indicated. Structure and `enum` types are described in scha_calls(3HA).

Tag Arguments

Macros that name resource properties are listed below. The value of the property of the resource is output. The `SCHA_RESOURCE_STATE`, `SCHA_STATUS`, `SCHA_NUM_RG_RESTARTS`, and `SCHA_NUM_RESOURCE_RESTARTS` properties refer to the value on the node where the command is executed (see r_properties(5)).

Extension properties

These properties are declared in the Resource Type Registration (RTR) file of the resource's type. The implementation of the resource type defines these properties.

`SCHA_AFFINITY_TIMEOUT`

The output argument type is `int*`.

`SCHA_ALL_EXTENSIONS`

The output argument type is `scha_str_array_t*`.

`SCHA_BOOT_TIMEOUT`

The output argument type is `int*`.

`SCHA_CHEAP_PROBE_INTERVAL`

The output argument type is `int*`.

`SCHA_EXTENSION`

The output argument type is `scha_extprop_value_t*`.

When a user requests the value of this property on a node or zone for which an explicit value has not been assigned, the default value that is declared in the RTR file is returned. See the rt_reg(4) man page.

`SCHA_EXTENSION_NODE`

The output argument type is `scha_extprop_value_t*`.

When a user requests the value of this property on a node or zone for which an explicit value has not been assigned, the default value that is declared in the RTR file is returned. See the rt_reg(4) man page.

`SCHA_FAILOVER_MODE`

The output argument type is `scha_failover_mode_t*`.

`SCHA_FINI_TIMEOUT`

The output argument type is `int*`.

`SCHA_GROUP`

The output argument type is `char**`.

`SCHA_INIT_TIMEOUT`

The output argument type is `int*`.

`SCHA_LOAD_BALANCING_POLICY`

The output argument type is `char**`.

`SCHA_LOAD_BALANCING_WEIGHTS`

The output argument type is `scha_str_array_t**`.

`SCHA_MONITOR_CHECK_TIMEOUT`
The output argument type is `int*`.

`SCHA_MONITOR_START_TIMEOUT`
The output argument type is `int*`.

`SCHA_MONITOR_STOP_TIMEOUT`
The output argument type is `int*`.

`SCHA_MONITORED_SWITCH`
The output argument type is `scha_switch_t*`.

`SCHA_MONITORED_SWITCH_NODE`
The output argument type is `scha_switch_t*`.

`SCHA_NETWORK_RESOURCES_USED`
The output argument type is `scha_str_array_t**`.

`SCHA_NUM_RESOURCE_RESTARTS`
The output argument type is `int*`.

`SCHA_NUM_RESOURCE_RESTARTS_ZONE`
The output argument type is `int*`.

`SCHA_NUM_RG_RESTARTS`
The output argument type is `int*`.

`SCHA_NUM_RG_RESTARTS_ZONE`
The output argument type is `int*`.

`SCHA_ON_OFF_SWITCH`
The output argument type is `scha_switch_t*`.

`SCHA_ON_OFF_SWITCH_NODE`
The output argument type is `scha_switch_t*`.

`SCHA_PORT_LIST`
The output argument type is `scha_str_array_t**`.

`SCHA_POSTNET_STOP_TIMEOUT`
The output argument type is `int*`.

`SCHA_PRENET_START_TIMEOUT`
The output argument type is `int*`.

`SCHA_R_DESCRIPTION`
The output argument type is `char**`.

`SCHA_RESOURCE_DEPENDENCIES`
The output argument type is `scha_str_array_t**`.

`SCHA_RESOURCE_DEPENDENCIES_Q`
The output argument type is `scha_str_array_t**`.

The "Q" version of the SCHA_RESOURCE_DEPENDENCIES tag also returns the scope, or qualifier, if any, that was declared for the resource dependency. The {LOCAL_NODE}, {ANY_NODE}, and {FROM_RG_AFFINITIES} qualifiers are described in the r_properties(5) man page.

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART
The output argument type is scha_str_array_t**.

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART_Q
The output argument type is scha_str_array_t**.

The "Q" version of the SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART tag also returns the scope, or qualifier, if any, that was declared for the resource dependency. The {LOCAL_NODE}, {ANY_NODE}, and {FROM_RG_AFFINITIES} qualifiers are described in the r_properties(5) man page.

SCHA_RESOURCE_DEPENDENCIES_RESTART
The output argument type is scha_str_array_t**.

SCHA_RESOURCE_DEPENDENCIES_RESTART_Q
The output argument type is scha_str_array_t**.

The "Q" version of the SCHA_RESOURCE_DEPENDENCIES_RESTART tag also returns the scope, or qualifier, if any, that was declared for the resource dependency. The {LOCAL_NODE}, {ANY_NODE}, and {FROM_RG_AFFINITIES} qualifiers are described in the r_properties(5) man page.

SCHA_RESOURCE_DEPENDENCIES_WEAK
The output argument type is scha_str_array_t**.

SCHA_RESOURCE_DEPENDENCIES_WEAK_Q
The output argument type is scha_str_array_t**.

The "Q" version of the SCHA_RESOURCE_DEPENDENCIES_WEAK tag also returns the scope, or qualifier, if any, that was declared for the resource dependency. The {LOCAL_NODE}, {ANY_NODE}, and {FROM_RG_AFFINITIES} qualifiers are described in the r_properties(5) man page.

SCHA_RESOURCE_PROJECT_NAME
The output argument type is char**.

SCHA_RESOURCE_STATE
The output argument type is scha_rsstate_t*.

SCHA_RESOURCE_STATE_NODE
The output argument type is scha_rsstate_t*.

SCHA_RETRY_COUNT
The output argument type is int*.

`SCHA_RETRY_INTERVAL`
The output argument type is `int*`.

`SCHA_SCALABLE`
The output argument type is `boolean_t*`.

`SCHA_START_TIMEOUT`
The output argument type is `int*`.

`SCHA_STATUS`
The output argument type is `scha_status_value_t**`.

`SCHA_STATUS_NODE`
The output argument type is `scha_status_value_t**`.

`SCHA_STOP_TIMEOUT`
The output argument type is `int*`.

`SCHA_THOROUGH_PROBE_INTERVAL`
The output argument type is `int*`.

`SCHA_TYPE`
The output argument type is `char**`.

`SCHA_TYPE_VERSION`
The output argument type is `char**`.

`SCHA_UDP_AFFINITY`
The output argument type is `boolean_t*`.

`SCHA_UPDATE_TIMEOUT`
The output argument type is `int*`.

`SCHA_VALIDATE_TIMEOUT`
The output argument type is `int*`.

`SCHA_WEAK_AFFINITY`
The output argument type is `boolean_t*`.

Macros that name resource type properties are listed below. The value of the property of the resource's type is output. For descriptions of resource type properties, see rt_properties(5).

`SCHA_API_VERSION`
The output argument type is `int*`.

`SCHA_BOOT`
The output argument type is `char**`.

`SCHA_FAILOVER`
The output argument type is `boolean_t*`.

`SCHA_FINI`
The output argument type is `char**`.

`SCHA_INIT`
The output argument type is `char**`.

`SCHA_INIT_NODES`
The output argument type is `scha_initnodes_flag_t*`.

`SCHA_INSTALLED_NODES`
The output argument type is `scha_str_array_t**`.

`SCHA_MONITOR_CHECK`
The output argument type is `char**`.

`SCHA_MONITOR_START`
The output argument type is `char**`.

`SCHA_MONITOR_STOP`
The output argument type is `char**`.

`SCHA_PKGLIST`
The output argument type is `scha_str_array_t**`.

`SCHA_POSTNET_STOP`
The output argument type is `char**`.

`SCHA_PRENET_START`
The output argument type is `char**`.

`SCHA_RT_BASEDIR`
The output argument type is `char**`.

`SCHA_RT_DESCRIPTION`
The output argument type is `char**`.

`SCHA_RT_SYSTEM`
The output argument type is `boolean_t*`.

`SCHA_RT_VERSION`
The output argument type is `char**`.

`SCHA_SINGLE_INSTANCE`
The output argument type is `boolean_t*`.

`SCHA_START`
The output argument type is `char**`.

`SCHA_STOP`
The output argument type is `char**`.

`SCHA_UPDATE`
The output argument type is `char**`.

`SCHA_VALIDATE`
The output argument type is `char**`.

Return Values These functions return the following values:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	The function succeeded.

See scha_calls(3HA) for a description of other error codes.

Examples **EXAMPLE 1** Using the `scha_resource_get()` Function

The following example uses `scha_resource_get()` to get the value of the `Retry_count` property of a resource, and the value of the extension property named `Loglevel`.

```
main() {
  #include <scha.h>

  scha_err_t err;
  int *retry_count_out;
  scha_extprop_value_t *loglevel_out;
  scha_resource_t handle;

/* a configured resource */
  char * resource_name = "example_R";
/* resource group containing example_R */
  char * group_name = "example_RG";

  err = scha_resource_open(resource_name, group_name, &handle);

  err = scha_resource_get(handle, SCHA_RETRY_COUNT, &retry_count_out);

/* Given extension property must be defined in resourcetype RTR file. */
  err = scha_resource_get(handle, SCHA_EXTENSION, "LogLevel", &loglevel_out);

  err = scha_resource_close(handle);

  printf("The retry count for resource %s is %d\n", resource_name,
         retry_count_out);

  printf("The log level for resource %s is %d\n", resource_name,
         loglevel_out->val.val_int);
}
```

Files

`</usr/cluster/include/scha.h>`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscdev`
Interface Stability	Evolving

See Also scha_resource_get(1HA), scha_calls(3HA), scha_strerror(3HA), `attributes`(5), r_properties(5), rt_properties(5), rt_reg(4)

Name scha_resource_setstatus, scha_resource_setstatus_zone – set resource status functions

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

scha_err_t scha_resource_setstatus(const char *rname, const char *rgname,
     scha_rsstatus_t status, const char *status_msg);

scha_err_t scha_resource_setstatus_zone(const char *rname, const char *rgname,
     const char *zonename, scha_rsstatus_t status, const char *status_msg);
```

Description The `scha_resource_setstatus()` and `scha_resource_setstatus_zone()` functions set the `Status` and `Status_msg` properties of a resource that is managed by the Resource Group Manager (RGM). A resource's monitor uses these functions to indicate the resource's state as perceived by the monitor.

Use the `scha_resource_setstatus()` and `scha_resource_setstatus_zone()` functions only for resource types whose `Global_zone` property is set to `TRUE`. These functions are not needed if the `Global_zone` property is set to `FALSE`. For more information, see the rt_properties(5) man page.

The *rname* argument names the resource whose status is to be set.

The *rgname* argument is the name of the resource group that contains the resource.

The *zonename* argument is the name of the non-global zone in which the resource group is configured to run. If the `Global_zone` property is set to `TRUE`, methods execute in the global zone even if the resource group that contains the resource runs in a non-global zone.

The *status* argument is an `enum` value of type `scha_rsstatus_t`: `SCHA_RSSTATUS_OK`, `SCHA_RSSTATUS_OFFLINE`, `SCHA_RSSTATUS_FAULTED`, `SCHA_RSSTATUS_DEGRADED`, or `SCHA_RSSTATUS_UNKNOWN`.

The *status_msg* argument is the new value for the `Status_msg` property. The *status_msg* argument can be `NULL`.

A successful call to the `scha_resource_setstatus()` or `scha_resource_setstatus_zone()` function causes the `Status` and `Status_msg` properties of the resource to be updated with the supplied values. The update of the resource status is logged in the cluster system log and is accessible by cluster administration tools.

Return Values The `scha_resource_setstatus()` and `scha_resource_setstatus_zone()` functions return the following values:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	The function succeeded.

See scha_calls(3HA) for a description of other error codes.

Examples

EXAMPLE 1 Using the `scha_resource_setstatus_zone()` Function

```
#include <scha.h>

scha_err_t err_code;
const char *rname = "example_R";
const char *rgname = "example_RG";

err_code = scha_resource_setstatus_zone(rname, rgname,
           SCHA_RSSTATUS_OK, "No problems");
```

Files

`/usr/cluster/include/scha.h`	Include file
`/usr/cluster/lib/libscha.so`	Library

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also

scha_resource_setstatus(1HA), scha_calls(3HA), scha_strerror(3HA), `attributes`(5), rt_properties(5)

Name scha_resourcetype_open, scha_resourcetype_close, scha_resourcetype_get – resource type information access functions.

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

scha_err_t scha_resourcetype_open(const char *rtname, scha_resourcetype_t *handle);

scha_err_t scha_resourcetype_close(scha_resourcetype_t handle);

scha_err_t scha_resourcetype_get(scha_resourcetype_t handle, const char *tag...);
```

Description You use the `scha_resourcetype_open()`, `scha_resourcetype_get()`, and `scha_resourcetype_close()` functions to access information about a resource type that is used by the Resource Group Manager (RGM) cluster facility.

`scha_resourcetype_open()` initializes access of the resource type and returns a handle to be used by `scha_resourcetype_get()`.

The *rtname* argument of `scha_resourcetype_open()` names the resource type to be accessed.

The *handle* argument is the address of a variable to hold the value returned from the function call.

`scha_resourcetype_get()` accesses resource type information as indicated by the *tag* argument. The *tag* argument should be a string value defined by a macro in the `<scha_tags.h>` header file. Arguments following the tag depend on the value of *tag*.

An additional argument following the tag may be needed to indicate a cluster node from which the information is to be retrieved, or other information specific to the tag. The last argument in the argument list is to be of a type suitable type to hold the information indicated by *tag*. This is the "out" argument for the resource type information. No value is returned for the out parameter if the function fails. Memory that is allocated to hold information returned by `scha_resourcetype_get()` remains intact until `scha_resourcetype_close()` is called on the handle used for `scha_resourcetype_get()`.

`scha_resourcetype_close()` takes a *handle* argument returned from a previous call to `scha_resourcetype_open()`. It invalidates the handle and frees memory allocated to return values to `scha_resourcetype_get()` calls that were made with the handle. Note that, memory, if needed to return a value, is allocated for each "get" call. Space allocated to return a value in one call will not be overwritten and reused by subsequent calls.

Macros defined in `<scha_tags.h>` that may be used as *tag* arguments to `scha_resourcetype_get()` follow. The type of the output argument and any additional arguments are indicated. Structure and enum types are described in scha_calls(3HA).

`optag` Arguments The following macros name resource type properties. The value of the named property of the resource's type is output.

Note – *optag* arguments, such as `SCHA_API_VERSION` and `SCHA_BOOT`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify *optag* arguments.

`SCHA_API_VERSION`
The output argument is of type `int*`.

`SCHA_BOOT`
The output argument is of type `char **`.

`SCHA_FAILOVER`
The output argument is of type `boolean_t *`.

`SCHA_FINI`
The output argument is of type `char **`.

`SCHA_GLOBALZONE`
The output argument is of type `boolean_t *`.

`SCHA_INIT`
The output argument is of type `char **`.

`SCHA_INIT_NODES`
The output argument is of type `scha_initnodes_flag_t *`.

`SCHA_INSTALLED_NODES`
The output argument is of type `scha_str_array_t **`.

`SCHA_IS_LOGICAL_HOSTNAME`
The output argument is of type `boolean_t *`.

`SCHA_IS_SHARED_ADDRESS`
The output argument is of type `boolean_t *`.

`SCHA_MONITOR_CHECK`
The output argument is of type `char **`.

`SCHA_MONITOR_START`
The output argument is of type `char **`.

`SCHA_MONITOR_STOP`
The output argument is of type `char **`.

`SCHA_PER_NODE`
The output argument is of type `boolean_t *`.

`SCHA_PKGLIST`
The output argument is of type `scha_str_array_t **`.

`SCHA_POSTNET_STOP`
The output argument is of type `char **`.

`SCHA_PRENET_START`
The output argument is of type `char **`.

`SCHA_PROXY`
The output argument is of type `boolean_t *`.

`SCHA_RESOURCE_LIST`
The output argument is of type `scha_str_array_t**`.

`SCHA_RT_BASEDIR`
The output argument is of type `char **`.

`SCHA_RT_DESCRIPTION`
The output argument is of type `char **`.

`SCHA_RT_SYSTEM`
The output argument is of type `boolean_t *`.

`SCHA_RT_VERSION`
The output argument is of type `char **`.

`SCHA_SINGLE_INSTANCE`
The output argument is of type `boolean_t *`.

`SCHA_START`
The output argument is of type `char **`.

`SCHA_STOP`
The output argument is of type `char **`.

`SCHA_UPDATE`
The output argument is of type `char **`.

`SCHA_VALIDATE`
The output argument is of type `char **`.

Return Values These functions return the following values:

`0`	The function succeeded.
nonzero	The function failed.

Errors

`SCHA_ERR_NOERR`	The function succeeded.

See the scha_calls(3HA) man page for a description of other error codes.

Files

`/usr/cluster/include/scha.h`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Evolving

See Also scha_resource_get(1HA), scha_calls(3HA), scha_strerror(3HA), scha_strerror_i18n(3HA), `attributes`(5), rt_properties(5)

Name scha_strerror, scha_strerror_i18n – generate error message from error code

Synopsis

```
cc [flags...] -I /usr/cluster/include file -L /usr/cluster/lib  -l  scha
#include <scha.h>

char *scha_strerror(scha_err_t error_code);

char *scha_strerror_i18n(scha_err_t error_code);
```

Description The `scha_strerror()` and `scha_strerror_i18n()` functions generate a short string that describes the error from the given `scha_err_t` error code. Strings that are returned by `scha_strerror()` are displayed in English. Strings that are returned by `scha_strerror_i18n()` are displayed in the native language that is specified by the `LC_MESSAGES` locale category. See `setlocale`(3C).

Parameters The following parameters are supported:

error_code Error code from which the short string that describes the error is generated.

Examples **EXAMPLE 1** Using the `scha_strerror_i18n()` Function

```
sample()
{
      scha_err_t  err;

      /* resource group containing example_R */
      char * resource_group = "example_RG";

      /* a configured resource */
      char * resource_name = "example_R";

      err = scha_control(SCHA_GIVEOVER, resource_group, resource_name);

      if (err != SCHA_ERR_NOERR) {
          syslog(LOG_ERR, "scha_control GIVEOVER failed: %s",
             scha_strerror_i18n(err));
      }
}
```

Files

`/usr/cluster/include/scha.h`	Include file
`/usr/cluster/lib/libscha.so`	Library

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also scha_calls(3HA), `setlocale`(3C), `syslog`(3C), `attributes`(5)

REFERENCE

SC32 4

Name clusters – cluster names database

Synopsis

```
/etc/clusters
```

Description The `clusters` file contains information regarding the known clusters in the local naming domain. For each cluster a single line should be present with the following information:

```
clustername       whitespace-delimited list of hosts
```

Expansion is recursive if a name on the right hand side is tagged with the expansion marker: "*".

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment. Characters up to the end of the line are not interpreted by routines which search the file.

Cluster names may contain any printable character other than an upper case character, a field delimiter, NEWLINE, or comment character. The maximum length of a cluster name is 32 characters.

This information is used by Sun Cluster system administration tools, like cconsole(1M) to specify a group of nodes to administer. The names used in this database must be host names, as used in the hosts database.

The database is available from either NIS or NIS+ maps or a local file. Lookup order can be specified in the `/etc/nsswitch.conf` file. The default order is nis files.

Examples **EXAMPLE 1** A Sample `/etc/clusters` File

Here is a typical `/etc/clusters` file:

```
bothclusters        *planets *wine
planets             mercury venus
wine                 zinfandel merlot chardonnay riesling
```

Here is a typical `/etc/nsswitch.conf` entry:

```
clusters: nis files
```

Files

```
/etc/clusters
```

```
/etc/nsswitch.conf
```

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Uncommitted

See Also cconsole(1M), chosts(1M), serialports(4), `nsswitch.conf`(4), `attributes`(5)

Name commandlog – command log file

Synopsis `/var/cluster/logs/commandlog`

Description The `commandlog` ASCII text file contains records of selected Sun Cluster commands that are executed in a cluster. The logging of commands starts automatically when you set up the cluster and ends when you shut down the cluster.

Commands that are not logged in this file include those that display the configuration and current state of the cluster. Commands that are logged in this file include those that configure and change the current state of the cluster, as follows:

- `claccess`
- `cldevice`
- `cldevicegroup`
- `clinterconnect`
- `clnasdevice`
- `clnode`
- `clquorum`
- `clreslogicalhostname`
- `clresource`
- `clresourcegroup`
- `clresourcetype`
- `clressharedaddress`
- `clsnmphost`
- `clsnmpmib`
- `clsnmpuser`
- `cltelemetryattribute`
- `cluster`
- `scconf`
- `scdidadm`
- `scdpm`
- `scgdevs`
- `scrgadm`
- `scshutdown`
- `scswitch`

Each record in the `commandlog` file contains the following information:

- Date and timestamp
- Host name from the which the command was executed
- Process ID of the command
- ID of the user who executed the command
- Command that the user executed, including all options and operands

Note – Command options are quoted in the `commandlog` file to enable you to copy, paste, and execute them in the shell.

- Exit status or signal of the executed command

By default, the `commandlog` file is regularly archived at the end of every week. Sun Cluster maintains up to eight previously archived `commandlog` files on each cluster node at any given time.

Examples

EXAMPLE 1 `/var/cluster/logs/commandlog` File

The following example shows the contents of a typical `/var/cluster/logs/commandlog` file:

```
11/11/2006 09:43:36 phys-schost-1 5758 root START - clrg add "app-sa-1"
11/11/2006 09:43:36 phys-schost-1 5758 root END 0
11/11/2006 09:43:36 phys-schost-1 5760 root START - clrg set -y
"RG_description=Department Shared Address RG" "app-sa-1"
11/11/2006 09:43:37 phys-schost-1 5760 root END 0
11/11/2006 09:44:15 phys-schost-1 5810 root START - clrg online "app-sa-1"
11/11/2006 09:44:15 phys-schost-1 5810 root END 0
11/11/2006 09:44:19 phys-schost-1 5222 root END -20988320
12/02/2006 14:37:21 phys-schost-1 5542 jbloggs START - clrg -c -g "app-sa-1"
-y "RG_description=Joe Bloggs Shared Address RG"
12/02/2006 14:37:22 phys-schost-1 5542 jbloggs END 0
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsczu
Interface Stability	Evolving

See Also

scha_control(1HA), scha_resource_setstatus(1HA), scconf(1M), scdidadm(1M), scdpm(1M), scgdevs(1M), scrgadm(1M), scshutdown(1M), scswitch(1M), `attributes`(5)

Name rt_reg – resource type registration (RTR) file

Description The resource type registration (RTR) file describes a resource type. Resource types represent highly-available or scalable services that run under the control of the Resource Group Manager (RGM) cluster facility. The file is part of a resource type implementation and is used as an input file for the scrgadm(1M) command to register the resource type in a cluster configuration. Registering the resource type is a prerequisite to creating resources of that type to run in the cluster. By convention, the RTR file resides in the `/opt/cluster/lib/rgm/rtreg` directory.

A RTR file declares the resource type properties and resource properties of a resource type. The file is divided into two parts, the declaration of resource type properties, and of resource properties. Note that recognition of property names is not case sensitive.

The resource type property declarations provide the information on the resource type implementation, such as paths to the callback methods that are to be invoked by the RGM to control resources of the type. Most resource type properties have fixed values set in the `rt_reg` file. These properties are inherited by all resources of the type.

A resource type implementor can also customize and extend the administrative view of resource properties. There are two kinds of resource properties that can have entries in the second part of an `rt_reg` file: system defined properties and extension properties.

System-defined resource properties have predetermined types and semantics. The `rt_reg` file can be used to set attributes such as default, minimum and maximum values for system defined resource properties. The `rt_reg` file can also be used to declare extension properties that are defined entirely by the resource type implementation. Extension properties provide a way for a resource type to add information to the configuration data for a resource that is maintained and managed by the cluster system.

The `rt_reg` file can set default values for resource properties, but the actual values are set in individual resources. The properties in the `rt_reg` file can be variables that can be set to different values and adjusted by the cluster administrator.

Resource Type Property Declarations The resource type property declarations consist of a number of property value assignments.

```
PROPERTY_NAME = "Value";
```

See the rt_properties(5) man page for a list of the resource type properties you can declare in the `rt_reg` file. Since most properties have default values or are optional, the only declarations that are essential in a RTR file are the type name, the paths to the `START` and `STOP` callback methods, and RT_version.

Note that the first property in the file must be the `Resource_type` property.

Starting in Sun Cluster 3.1, a resource type name is of the form

```
vendor_id.rtname:version
```

The three components of the resource type name are properties specified in the RTR file as *Vendor_id*, *Resource_type*, and *RT_version*; the `scrgadm` command inserts the period and colon delimiters. Although optional, the *Vendor_id* prefix is recommended to distinguish betweentwo registration files of the same name provided by different vendors. To ensure that the *Vendor_id* is unique, the recommended approach is to use the stock symbol for the company creating the resource type.

Resource type names created prior to Sun Cluster 3.1 continue to be of the form:

```
vendor_id.rtname
```

Resource Property Declarations

Resource property declarations consist of a number of entries, each entry being a bracketed list of attribute value assignments. The first attribute in the entry must be the resource property name.

System-defined properties have predetermined type and description attributes and so these attributes cannot be redeclared in the `rt_reg` file. Range restrictions, a default value, and constraints on when the value can be set by the administrator can be declared for system defined properties.

Attributes that can be set for system-defined properties are listed in the property_attributes(5) man page. Attributes not available for system-defined properties are noted as such in the table.

System-defined properties that can have entries in the `rt_reg` file are listed in the r_properties(5) man page. The following is a sample entry for the system defined `RETRY_COUNT` resource property.

```
{
  PROPERTY = RETRY_COUNT;
  MIN=0;
  MAX=10;
  DEFAULT=2;
  TUNABLE = ANYTIME;
}
```

Entries for extension properties must indicate a type for the property. Attributes that can be set for extension properties are listed in the property_attributes(5) man page.

The following is a sample entry for an extension property named "`ConfigDir`" that is of string type. The `TUNABLE` attribute indicates that the cluster administrator can set the value of the property when a resource is created.

```
{
  PROPERTY = ConfigDir;
  EXTENSION;
  STRING;
  DEFAULT="/";
  TUNABLE = AT_CREATION;
}
```

Usage An `rt_reg` file is an ASCII text file. It can include comments describing the contents of the file. The contents are the two parts described above, with the resource type property list preceding the resource property declarations.

White space can be blanks, tabs, newlines, or comments. White space can exist before or after tokens. Blanks and the pound sign (#) are not considered to be white space when found in quoted value tokens. White space separates tokens but is otherwise ignored.

Comments begin with # and end with the first newline encountered, inclusively.

Directives begin with #$ and end with the first newline encountered, inclusively. Directives must appear in the RTR file between the resource type property declaration section and the resource property declaration section. Directives inserted in any other location in the RTR file will produce parser errors. The only valid directives are `#$upgrade` and `#$upgrade_from`. Any other directive will produce parser errors.

Tokens are property names, property values, and the following:

`{ }` Encloses parameter table properties

`;` Terminates properties and attributes

`=` Separates property names and property values or attribute names and attribute values

`,` Separates values in a value list

The recognition of property-name keywords in the file is case insensitive.

Properties and attributes have one of three formats.

```
<property-name> = <property-value>;
<property-name>;
<property-name> = <property-value> [, <property-value>];
```

In the format above, the square brackets, `[ ]`, enclose optional items. That is, the property value can be a single `<property-value>` or a list of two or more `<property-value>`s separated by commas.

The first property in the property list must be the simple resource type name.

Boolean properties and attributes have the following syntax:

```
<boolean-property-name>;
<boolean-property-name> = TRUE;
<boolean-property-name> = FALSE;
```

The first and second forms both set the `<boolean-property-name>` to `TRUE`.

The only property name taking a list for its value is PKGLIST. An example is:

```
PKGLIST = SUNWsczu, SUNWrsm;
```

Resource type property names are listed in the rt_properties(5) man page. System-defined properties are listed in the r_properties(5) man page.

Resource declarations consist of any number of entries, each being a bracketed list of resource property attributes.

```
{<attribute-value-list>}
```

Each attribute-value-list consists of attribute values for a resource property, in the same syntax used for property values, with the addition of the two type-attribute formats.

```
<type-attribute-value>;
<enum-type-attribute> { <enum-value> [ , <enum-value> ] };
```

The <type-attribute-value> syntax declares the data type of the extension property to have the value <type-attribute-value>. It differs from the first format of the <boolean-property-name>, which defines the property named by <boolean-property-name> to have the value TRUE.

For example, the TUNABLE attribute can have one of the following values: FALSE or NONE, AT_CREATION, TRUE or ANYTIME, and WHEN_DISABLED. When the TUNABLE attribute uses the syntax:

```
TUNABLE;
```

it gets the value of ANYTIME.

Grammar

The following is a description of the syntax of the rt_reg file with a BNF-like grammar. Non-terminals are in lower case, and terminal keywords are in upper case, although the actual recognition of keywords in the rt_reg file is case insensitive. The colon (:) following a non-terminal at the beginning of a lines indicates a grammar production. Alternative right-hand-sides of a grammar production are indicated on lines starting with a vertical bar (|). Variable terminal tokens are indicated in angled brackets and comments are parenthesized. Other punctuation in the right-hand side of a grammar production, such as semi-colon (;), equals sign (=), and angled brackets ({}) are literals.

A comment has the form:

```
COMMENT    : # <anything but NEWLINE> NEWLINE
```

Comments may appear after any token. Comments are treated as white-space.

```
    rt_reg_file    :  Resource_type = value ; proplist upgradesect paramtable
```

```
proplist    :  (NONE: empty)
| proplist rtproperty

rtproperty    : rtboolean_prop ;
| rtvalue_prop ;

rtboolean_prop    : SINGLE_INSTANCE
| FAILOVER | RT_SYSTEM

rtvalue_prop    : rtprop = value
| PKGLIST = valuelist

rtprop    : RT_BASEDIR
| RT_VERSION
| API_VERSION
| INIT_NODES
| START
| STOP
| VALIDATE
| UPDATE
| INIT
| FINI
| BOOT
| MONITOR_START
| MONITOR_STOP
| MONITOR_CHECK
| PRENET_START
| POSTNET_STOP
| RT_DESCRIPTION
| VENDOR_ID
| rtboolean_prop (booleans may have explicit assignments.)

value    : <contiguous-non-ws-non-;-characters>
| "<anything but quote>"
| TRUE
| FALSE
| ANYTIME
| WHEN_DISABLED
| AT_CREATION
| RG_PRIMARIES
| RT_INSTALLED_NODES
|  (NONE: Empty value)

valuelist    : value
| valuelist , value

upgradesect : (empty)
```

```
| #$UPGRADE upgradelist

upgradelist :  (empty)
| upgradelist #$UPGRADE_FROM rt_version upgtunability

upgtunability : ANYTIME
| AT_CREATION
| WHEN_DISABLED
| WHEN_OFFLINE
| WHEN_UNMANAGED
| WHEN_UNMONITORED

paramtable    :  (empty)
| paramtable parameter

parameter    :  { pproplist }

pproplist    : PROPERTY = value ;  (property name must come first)
| pproplist pproperty

pproperty    : pboolean_prop ;
| pvalue_prop ;
| typespec ;

pvalue_prop    : tunable_prop
| pprop = value
| pprop =  (NONE: no value setting)
| DEFAULT = valuelist

pprop    : DESCRIPTION
| MIN
| MAX
| MINLENGTH
| MAXLENGTH
| ARRAY_MINSIZE
| ARRAY_MAXSIZE
| pboolean_prop

tunable_prop    : TUNABLE
| TUNABLE = AT_CREATION
| TUNABLE = ANYTIME
| TUNABLE = WHEN_DISABLED
| TUNABLE = TRUE
| TUNABLE = FALSE
| TUNABLE = NONE

typespec    : INT
```

```
    | BOOLEAN
    | STRING
    | STRINGARRAY
    | ENUM { valuelist }
```

Examples

EXAMPLE 1 A Sample Registration File

The following is the registration file for a simple example resource type.

```
#
# Registration information for example resource type
#

Resource_type = example_RT;
Vendor_id = SUNW;
RT_Version = 2.0
RT_Basedir= /opt/SUNWxxx;
START = bin/example_service_start;
STOP  = bin/example_service_stop;
Pkglist = SUNWxxx;

#$upgrade
#$upgrade_from "1.0" when_unmonitored

#
# Set range and defaults for method timeouts and Retry_count.
#
{ Property = START_TIMEOUT;  Tunable; MIN=60; DEFAULT=300; }
{ Property = STOP_TIMEOUT;  Tunable; MIN=60; DEFAULT=300; }
{ Property = Retry_count;  Tunable; MIN=1; MAX=20; DEFAULT=10; }

#
# An extension property that can be set at resource creation
#
{ Property = LogLevel;
  Extension;
  enum { OFF, TERSE, VERBOSE };
  Default = TERSE;
  Tunable = AT_CREATION;
  Description = "Controls the detail of example_service logging";
}
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also scrgadm(1M), `attributes`(5), rt_properties(5), r_properties(5), property_attributes(5)

Sun Cluster Data Services Developer's Guide for Solaris OS

Name scqsd.conf – startup file for the quorum server

Synopsis `/etc/scqsd/scqsd.conf`

Description The `/etc/scqsd/scqsd.conf` file contains information that allows the `clquorumserver` command to manage quorum server instances on a machine. Each line in this file has the following format:

full-path-to-quorum-server-binary `[-i` *instancename*`] [-p` *portname*`] [-d` *quorumdirectory*`]`

Lines beginning with a number sign (#) are treated as comments and ignored. Every line is executed as specified in the file.

See scqsd(1M) for information about specific options.

Examples **EXAMPLE 1** Initial scqsd.conf File

The following screen shows the default contents of the configuration file.

```
#Copyright 2005 Sun Microsystems, Inc. All rights reserved.
#Use is subject to license terms.
#
#ident "@(#)scqsd.conf 1.4  05/10/10
#
#This is the startup for for Quorum Server daemons.
#Each line in the file starts up an instance of the quorum server
#daemon. The command must have the following format:
#
#full-parth-to-quorum-server-binary 9-i instancename] [-p [port] \
#  [-d quorumdirectory]
#
#The minimum requirement to start multiple quorum servers is to
#speciry a unique port and a unique quorum directory for
#each quorum server instance.
#
#To configure more instances, add commands to this file.
#Lines beginning with a # are treated as comments and ignored.
#
/usr/cluster/lib/sc/scqsd -d /var/scqsd -p 9000
```

See Also scqsd(1M), clquorumserver(1CL)

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscqs`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Stability	evolving

Name serialports – name to serial port database

Synopsis

```
/etc/serialports
serialports NIS or NIS+ maps
```

Description The `serialports` database maps a name to a server name and TCP port number that represents the serial port connected to the specified terminal server host. The database is typically used to map host names to their consoles, but may also be used to provide access to printers, modems, and the like. The mapping is used when the service is being provided by a network based terminal concentrator. For each name a single line should be present with the following information:

```
host-name    concentrator-hostname tcp-port-number
```

Items are separated by any number of blanks or TAB characters. A pound sign (#)indicates the beginning of a comment. Characters between the pound sign and the end of the line are not interpreted by routines that search the file.

This information is used by the `cconsole` or `cssh` command to establish connection to a group of consoles of a cluster of network hosts. The names that are used in this database must be host names, as used in the `/etc/inet/hosts` database.

To support Secure Shell connections to node consoles through the `cssh` command, specify in the `/etc/serialports` file the name of the console-access device and the Secure Shell port number for each node. If you use the default Secure Shell configuration on the console-access device, specify port number 22.

For E10000 nodes, the entries are different. This is because E10000 uses `netcon` for console purposes, which operates over a network and executes on the SSP. The following is the generic format for the entry.

hostname SSPname `23`

The database is available from either the NIS or NIS+ maps or a local file. Lookup order is specified by the `serialports` entry in the `/etc/nsswitch.conf` file, if present. If no search order is specified, the default order is `nis files`.

Examples **EXAMPLE 1** A Sample `/etc/serialports` File

The following is an example `/etc/serialports` file:

```
# Network host to port database

#  NFS server cluster
mercury        planets-tc    5001
venus        planets-tc    5002

# E10000 server cluster
cashews         nuts-ssp-1 23
```

EXAMPLE 1 A Sample `/etc/serialports` File *(Continued)*

```
pecans          nuts-ssp-2 23
```

EXAMPLE 2 A Sample `/etc/nsswitch.conf` File Entry

The following is a typical `/etc/nsswitch.conf` entry:

```
serialports: nis files
```

Files

`/etc/serialports`

`/etc/nsswitch.conf`

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscdev
Interface Stability	Uncommitted

See Also

cconsole(1M), chosts(1M), cports(1M), cssh(1M), clusters(4), `nsswitch.conf`(4), `attributes`(5)

REFERENCE

SC32 5

Name property_attributes – resource property attributes

Description The list below describes the resource property attributes that you can use to change system-defined properties or create extension properties.

You cannot specify `NULL` or the empty string ("") as the default value for `Boolean`, `Enum`, or `Int` types.

`Array_maxsize`
For `Stringarray` type, the maximum number of array elements permitted.

`Array_minsize`
For `Stringarray` type, the minimum number of array elements that is permitted.

`Default`
Indicates a default value for the property.

`Description`
A string annotation intended to be a brief description of the property. The description attribute cannot be set in the RTR file for system-defined properties.

`Enumlist`
For an `Enum` type, a set of string values permitted for the property.

`Extension`
If used, indicates that the RTR file entry declares an extension property defined by the resource type implementation. Otherwise, the entry is a system-defined property.

`Max`
For an `Int` type, the maximum value permitted for the property. Note that you cannot specify a maximum value for a method timeout.

`Maxlength`
For `String` and `Stringarray` types, the maximum string length that is permitted.

`Min`
For an `Int` type, the minimal value permitted for the property. Note that you cannot specify `Min=0` for a method timeout.

`Minlength`
For `String` and `Stringarray` types, the minimum string length permitted.

`Per_node`
If used, indicates that the extension property can be set on a per-node or a per-zone basis.

If you specify the `Per_node` property attribute in a type definition, you must specify a default value with the `Default` property attribute as well. Specifying a default value ensures that a value is returned when a user requests a per-node or per-zone property value on a node or zone to which an explicit value has not been assigned.

`Property`
The name of the resource property.

Property Type
Allowed types are: `String`, `Boolean`, `Int`, `Enum`, and `Stringarray`. You cannot set the type attribute in an RTR file entry for system-defined properties. The type determines acceptable property values and the type-specific attributes that are allowed in the RTR file entry. An `Enum` type is a set of string values.

`Tunable`
Indicates when the cluster administrator can set the value of this property in a resource. Can be set to `None` or `False` to prevent the administrator from setting the property. Values that allow administrator tuning are: `True` or `Anytime` (at any time), `At_creation` (only when the resource is created), or `When_disabled` (when the resource is offline).

The default is `True` (`Anytime`).

Examples

EXAMPLE 1 An `Int` Type Definition

An `Int` type definition might look like this:

```
{
         Property = Probe_timeout;
         Extension;
         Int;
         Default = 30;
         Tunable = Anytime;
         Description = "Time out value for the probe (seconds)";
}
```

EXAMPLE 2 A `Per_node` Type Definition

A `Per_node` type definition might look like this:

```
{
         Property = LogLevel;
         Extension;
         Enum { Off, Terse, Verbose };
         Default = Terse;
         Per_node;
         Tunable = At_creation;
         Description ="Controls the level of detail for logging";
}
```

If you specify the `PER_NODE` property attribute in a type definition, you must specify a default value with the `DEFAULT` property attribute as well.

See Also

clresource(1CL), clresourcegroup(1CL), clresourcetype(1CL), r_properties(5) rg_properties(5), rt_properties(5)

Name rg_properties – resource group properties

Description The following information describes the resource group properties that are defined by Sun Cluster.

Resource Group Properties and Descriptions

Note – Resource group property names, such as `Auto_start_on_new_cluster` and `Desired_primaries`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify resource group property names.

`Auto_start_on_new_cluster` (`boolean`)
This property controls whether the Resource Group Manager (RGM) starts the resource group automatically when a new cluster is forming. The default is `TRUE`.

If set to `TRUE`, the RGM attempts to start the resource group automatically to achieve `Desired_primaries` when all the nodes in the cluster are simultaneously rebooted.

If set to `FALSE`, the resource group does not start automatically when the cluster is rebooted. The resource group remains offline until the first time that the resource group is manually switched online by using the clresourcegroup(1CL) command or the equivalent graphical user interface command. After that, the resource group resumes normal failover behavior.

Default `TRUE`

Tunable Any time

`Desired_primaries` (`integer`)
The desired number of nodes or zones on which the resource group can run simultaneously.

The default is 1. The value of the `Desired_primaries` property must be less than or equal to the value of the `Maximum_primaries` property.

Default 1, see above

Tunable Any time

`Failback` (`boolean`)
A Boolean value that indicates whether to recalculate the set of nodes or zones where the resource group is online when the cluster membership changes. A recalculation can cause the RGM to bring the group offline on less preferred nodes or zones and online on more preferred nodes or zones.

Default `FALSE`

Tunable Any time

`Global_resources_used` (`string_array`)
Indicates whether cluster file systems are used by any resource in this resource group. Legal values that the administrator can specify are an asterisk (*) to indicate all global resources, and the empty string ("") to indicate no global resources.

Default All global resources

Tunable Any time

`Implicit_network_dependencies` (`boolean`)
A Boolean value that indicates, when `TRUE`, that the RGM should enforce implicit strong dependencies of non-network-address resources on network-address resources within the group. This means that the RGM starts all network-address resources before all other resources and stops network address resources after all other resources within the group. Network-address resources include the logical host name and shared address resource types.

In a scalable resource group, this property has no effect because a scalable resource group does not contain any network-address resources.

Default `TRUE`

Tunable Any time

`Maximum_primaries` (`integer`)
The maximum number of nodes or zones where the resource group might be online at the same time.

If the `RG_mode` property is `Failover`, the value of this property must be no greater than 1. If the `RG_mode` property is `Scalable`, a value greater than 1 is allowed.

Default 1, see above

Tunable Any time

`Nodelist` (`string_array`)
A list of nodes or zones where the group can be brought online in order of preference. These nodes or zones are known as the potential primaries or masters of the resource group.

To specify a non-global zone as an element of `Nodelist`, use the following syntax:

nodename:*zonename*

nodename is the name of the node where *zonename* is located. *zonename* is the name of the zone that you want to include in `Nodelist`. For example, to specify the non-global zone `zone-1`, which is located on the node `phys-schost-1`, you specify the following text:

```
phys-schost-1:zone-1
```

Default The list of all cluster nodes in arbitrary order

Tunable Any time

`Pathprefix` (`string`)
A directory in the cluster file system in which resources in the group can write essential administrative files. Some resources might require this property. Make `Pathprefix` unique for each resource group.

Default The empty string

Tunable Any time

`Pingpong_interval` (`integer`)
A non-negative integer value (in seconds) used by the RGM to determine where to bring the resource group online in the event of a reconfiguration or as the result of an `scha_control` giveover command or function being executed.

In the event of a reconfiguration, if the resource group fails more than once to come online within the past `Pingpong_interval` seconds on a particular node or zone (because the resource's `Start` or `Prenet_start` method exited nonzero or timed out), that node or zone is considered ineligible to host the resource group and the RGM looks for another master.

If a scha_control(1HA) command or scha_control(3HA) giveover is executed on a given node or zone by a resource, thereby causing its resource group to fail over to another node or zone, the first node or zone (on which `scha_control` was invoked) cannot be the destination of another `scha_control` giveover by the same resource until `Pingpong_interval` seconds have elapsed.

Default 3600 (one hour)

Tunable Any time

`Resource_list` (`string_array`)
The list of resources that are contained in the group. The administrator does not set this property directly. Rather, the RGM updates this property as the administrator adds or removes resources from the resource group.

Default No default

Tunable Never

`RG_affinities` (`string`)
The RGM is to try (1) to locate a resource group on a node or zone that is a current master of another given resource group (positive affinity) or (2) to locate a resource group on a node or zone that is not a current master of a given resource group (negative affinity).

You can set `RG_affinities` to the following strings:

- +, or weak positive affinity
- ++, or strong positive affinity
- +++, or strong positive affinity with failover delegation
- -, or weak negative affinity
- --, or strong negative affinity

For example, `RG_affinities=+RG2,--RG3` indicates that this resource group has a weak positive affinity for RG2 and a strong negative affinity for RG3.

Using `RG_affinities` is described in Chapter 2, "Administering Data Service Resources," in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS* in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*.

Default The empty string

Tunable Any time

`RG_dependencies` (`string_array`)
Optional list of resource groups that indicate a preferred ordering for bringing other groups online or offline on the same node or zone. The graph of all strong `RG_affinities` (positive and negative) together with `RG_dependencies` is not allowed to contain cycles.

For example, suppose that resource group RG2 is listed in the `RG_dependencies` list of resource group RG1. In other words, suppose that RG1 has a resource group dependency on RG2. The following list summarizes the effects of this resource group dependency:

- When a node or zone joins the cluster, `Boot` methods on that node or zone are not run on resources in RG1 until all `Boot` methods on that node or zone have completed on resources in RG2.
- If RG1 and RG2 are both in the `Pending_online` state on the same node or zone at the same time, the start methods (`Prenet_start` or `Start`) are not run on any resources in RG1 until all the resources in RG2 have completed their start methods.
- If RG1 and RG2 are both in the `Pending_offline` state on the same node or zone at the same time, the stop methods (`Stop` or `Postnet_stop`) are not run on any resources in RG2 until all the resources in RG1 have completed their stop methods.
- An attempt to switch the primaries of RG1 or RG2 fails if switching the primaries would leave RG1 online on any node or zone and RG2 offline on all nodes or zones.
- Setting the `Desired_primaries` property to a value that is greater than zero on RG1 is not permitted if `Desired_primaries` is set to zero on RG2.
- Setting the `Auto_start_on_new_cluster` property to `TRUE` on RG1 is not permitted if `Auto_start_on_new_cluster` is set to `FALSE` on RG2.

Default The empty list

Tunable Any time

`RG_description` (`string`)
A brief description of the resource group.

Default The empty string

Tunable Any time

`RG_is_frozen` (`boolean`)
A Boolean value that indicates whether a global device on which a resource group depends is being switched over. If this property is set to `TRUE`, the global device is being switched

over. If this property is set to FALSE, no global device is being switched over. A resource group depends on global devices as indicated by its Global_resources_used property.

You do not set the RG_is_frozen property directly. The RGM updates the RG_is_frozen property when the status of the global devices changes.

Default No default

Tunable Never

RG_mode (enum)
Indicates whether the resource group is a failover or a scalable group. If the value is Failover, the RGM sets the Maximum_primaries property of the group to 1 and restricts the resource group to being mastered by a single node or zone.

If the value of this property is Scalable, the RGM allows the Maximum_primaries property to be set to a value that is greater than 1. As a result, the group can be mastered by multiple nodes or zones simultaneously. The RGM does not allow a resource whose Failover property is TRUE to be added to a resource group whose RG_mode is Scalable.

If Maximum_primaries is 1, the default is Failover. If Maximum_primaries is greater than 1, the default is Scalable.

Default Depends on the value of Maximum_primaries

Tunable At creation

RG_project_name (string)
The Solaris project name (see projects(1)) that is associated with the resource group. Use this property to apply Solaris resource management features, such as CPU shares and resource pools, to cluster data services. When the RGM brings resource groups online, it launches the related processes under this project name for resources that do not have the Resource_project_name property set (see r_properties(5)). The specified project name must exist in the projects database(see projects(1) and *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

This property is supported starting with the Solaris 9 OS.

Note – Changes to this property take affect the next time that the resource is started.

Default The text string "default"

Tunable Any time

Valid value Any valid Solaris project name

RG_SLM_CPU_SHARES (integer)
The number of CPU shares associated with the resource group.

Note – You can only set the RG_SLM_CPU_SHARES property if RG_SLM_TYPE is set to automated. For more information, see the RG_SLM_TYPE property.

The maximum value for RG_SLM_CPU_SHARES is 65535. Zero is not an acceptable value for RG_SLM_CPU_SHARES because setting a share value to zero can lead to processes not being scheduled when the CPU is heavily loaded. Changes made to RG_SLM_CPU_SHARES while the resource group is online are taken into account dynamically.

Because RG_SLM_TYPE is set to automated, Sun Cluster creates a project(4) named SCSLM_*resourcegroup_name*, where *resourcegroup_name* is the name you give to the resource group. Each method of a resource that belongs to the resource group is executed in this project. In the Solaris 10 release, these projects are created in the resource group's zone, whether it is a global zone or a non-global zone.

The project SCSLM_*resourcegroup_name* has a project.cpu-shares value set to the RG_SLM_CPU_SHARES value. If the RG_SLM_CPU_SHARES property is not set, this project is created with a project.cpu-shares value of 1.

In the Solaris 10 release, when the RG_SLM_PSET_TYPE property is set to strong or weak, the value of RG_SLM_CPU_SHARES property is also used to compute the size of pset created (by convention, 100 shares are equivalent to one CPU). For more information, see the RG_SLM_PSET_TYPE property.

For information about processor sets, see *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

Default 1

Tunable Any time

RG_SLM_PSET_MIN (integer)

The minimum number of processors in the processor set in which the resource group executes. You can only use this property if the following are true:

- The operating system used is Solaris 10.
- RG_SLM_TYPE is set to automated.
- RG_SLM_PSET_TYPE is set to strong or weak. (See the RG_SLM_PSET_TYPE property.)
- The value of RG_SLM_PSET_MIN must be lower or equal to the value of the RG_SLM_CPU_SHARES divided by 100.

The maximum number of for RG_SLM_PSET_MIN is 655. The value of the RG_SLM_PSET_MIN property is used by Sun Cluster to computer the minimum size of processor sets.

Changes made to RG_SLM_CPU_SHARES and RG_SLM_PSET_MIN while the resource group is online are taken into account dynamically. However, if RG_SLM_PSET_TYPE is set to strong, and if there are not enough CPUs available to accommodate the change, the change requested for RG_SLM_PSET_MIN is not applied. In this case, a warning message is displayed. On next switchover, errors due to lack of CPUs can occur if there are not enough CPUs available to respect the values you configured for RG_SLM_PSET_MIN.

For information about processor sets, see *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

Default `0`

Tunable Any time

`RG_SLM_PSET_TYPE` (`string`)
Enables the creation of a dedicated processor set.

Possible values for `RG_SLM_PSET_TYPE` are `default`, `strong`, and `weak`.

You can set `RG_SLM_PSET_TYPE` to `strong` or `weak` if all of the following criteria are true:

- The operating system used is Solaris 10.
- The resource group is configured to execute only in a non-global zone.
- `RG_SLM_TYPE` is set to `automated`.

Possible values for `RG_SLM_PSET_TYPE` are `default`, `strong`, and `weak`.

For a resource group to execute as `strong` or `weak`, the resource group must be configured so there are only non-global zones in its node list.

The non-global zone must not be configured for a pool other than the default pool (`pool_default`). For information about zone configuration, see `zonecfg`(1M). This non-global zone must not be dynamically bound to a pool other than the default pool. For more information on pool binding, see `poolbind`(1M). These two pool conditions are verified only when the methods of the resources in the resource group are started.

The values `strong` and `weak` are mutually exclusive for resource groups that have the same zone in their node list. You cannot configure resource groups in the same zone so that some have `RG_SLM_PSET_TYPE` set to `strong` and others set to `weak`.

If `RG_SLM_PSET_TYPE` is set to `strong` or `weak` and the actions listed for `RG_SLM_TYPE` are set to `automated`, when the resource group is brought online, Sun Cluster does the following:

- Creates a pool and dynamically binds this pool to the non-global zone in which the resource group starts.
- Creates a processor set with a size between a minimum and maximum value.
  - The minimum value is the sum of `RG_SLM_PSET_MIN` values of all the resource groups online in the zone this resource group starts in, or 1 if that sum equals zero.
  - The maximum value is the sum of `RG_SLM_SPU_SHARES` values of all resource groups online in that zone, divided by 100, and rounded up to the immediate upper integer, or 1 if the result of the computation is zero.
- Associates the processor set to the pool.
- Sets `zone.cpu-shares` to the sum of `RG_SLM_CPU_SHARES` in all of the resource groups running in the zone.

If RG_SLM_PSET_TYPE is set to strong or weak, then the resource group is brought offline (more precisely when the STOP or POSTNET_STOP method of the resource group's first resource is executed), Sun Cluster destroys the processor set if there are no longer any resource groups online in the zone, destroys the pool, and binds the zone to the default pool (pool_default).

If RG_SLM_PSET_TYPE is set to strong, the resource group behaves the same as if RG_SLM_PSET_TYPE was set to strong. However, if there are not enough processors available to create the processor set, the pool is associated with the default processor set.

If RG_SLM_PSET_TYPE is set to strong and there are not enough processors available to create the processor set, an error is returned to the Resource Group Monitor (RGM), and the resource group is not started on that node or zone.

The order of priority for CPU allocation is defaultpsetmin minimum size has priority over strong, which has priority over weak. (For information about the defaultpsetmin property, see clnode(1CL).) However, this priority is not maintained when you try to increase the size of the default processor set by using the clnode command and there are not enough processors available.

If you assign a minimum number of CPUs to the default processor set by using the clnode command, the operation is done dynamically. If the number of CPUs that you specify is not available, Sun Cluster periodically retries to assign this number of CPUs, and subsequently smaller numbers of CPUs, to the default processor set until the minimum number of CPUs has been assigned. This action might destroy some weak processor sets, but does not destroy strong processor sets.

When a resource group with RG_SLM_PSET_TYPE configured as strong starts, it might destroy the processor sets associated with the weak processor sets if there are not enough CPU available on the node for both processor sets. In that case, the processes of the resource group running in the weak processor sets are associated with the default processor set.

To change a processor set from weak to strong or from strong to weak, you must first change the processor set to have RG_SLM_PSET_TYPE set to default.

If you set RG_SLM_PSET_TYPE to default, Sun Cluster creates a pool, SCSLM_pool_*zone_name*, but does not create a processor set. In this case, SCSLM_pool_*zone_name* is associated with the default processor set. The shares that are assigned to the zone are determined by the sum of the values that are set for RG_SLM_CPU_SHARES for all of the resource groups that are running in the zone.

If there are no longer any online resource groups configured for CPU control in a non-global zone, the CPU share value for the non-global zone takes the value of zone.cpu-shares found in the zone configuration. This parameter has a value of 1 by default. For more information about zone configuration, see zonecfg(1M).

For information about resource pools and processor sets, see *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

Default The text string "`default`"

Tunable Any time

`RG_SLM_TYPE` (`string`)

Enables you to control system resource usage, and automates some steps to configure the Solaris OS for system resource management. Possible values for `RG_SLM_TYPE` are `automated` and `manual`.

If `RG_SLM_TYPE` is set to `automated`, when the resource group is brought online, Sun Cluster does the following:

- Creates a project named `SCSLM_`*resourcegroup_name*. All methods in the resources in this resource group execute in this project. This project is created the first time a method of a resource in this resource group is executed on the node or zone.
- Sets the value of `project.cpu_shares` that is associated with the project to the value of `RG_SLM_CPU_SHARES`. The value of `project.cpu_shares` is 1 by default.
- In the Solaris 10 release, sets `zone.cpu-shares` to the sum of `RG_SLM_CPU_SHARES` of all the resource groups with `RG_SLM_TYPE` set to `automated` for the zone. The zone can be global or non-global. The non-global zone is bound to a Sun Cluster generated pool. Optionally, this Sun Cluster generated pool is associated with a Sun Cluster generated dedicated processor set if `RG_SLM_PSET_TYPE` is set to `weak` or `strong`. For information about dedicated processor sets, see the `RG_SLM_PSET_TYPE` property.

When `RG_SLM_TYPE` is set to `automated`, any action taken results in a message being logged.

If `RG_SLM_TYPE` is set to `manual`, the resource group executes in the project specified by the `RG_project_name` property.

For information about resource pools and processor sets, see *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

Note –

- Do not specify resource group names that exceed 58 characters. If a resource group name contains more than 58 characters, you cannot configure CPU control, that is, you cannot set the `RG_SLM_TYPE` property to `automated`.
- Refrain from including dashes (-) in resource group names. The Sun Cluster software replaces all dashes in resource group names with underscores (_) when it creates a project. For example, Sun Cluster creates the project named `SCSLM_rg_dev` for a resource group named `rg-dev`. If a resource group named `rg_dev` already exists, a conflict arises when Sun Cluster attempts to create the project for the resource group `rg-dev`.

Default `manual`

Tunable Any time

`RG_state` on each cluster node or zone (`enum`)
Set by the RGM to `Unmanaged`, `Online`, `Offline`, `Pending_online`, `Pending_offline`, `Error_stop_failed`, `Online_faulted`, or `Pending_online_blocked` to describe the state of the resource group on each cluster node or zone.

You cannot configure this property. However, you can indirectly set this property by using clresourcegroup(1CL) or by using the equivalent Sun Cluster graphical user interface command. A resource group can exist in an `Unmanaged` state when that group is not under the control of the RGM.

The following descriptions summarize each state.

Note – States apply to individual nodes or zones only, except the `Unmanaged` state, which applies across all nodes or zones. For example, a resource group might be `Offline` on node A, but `Pending_online` on node B.

`Error_stop_failed`	One or more resources within the resource group failed to stop successfully and are in the `Stop_failed` resource state. Other resources in the group might remain online or offline. This resource group is not permitted to start on any node or zone until the `Error_stop_failed` state is cleared. You must use an administrative command, such as `clresourcegroup clear`, to manually kill the `Stop_failed` resource and reset its state to `Offline`.
`Offline`	The resource group has been stopped on the node or zone. In other words, the stop methods (`Monitor_stop`, `Stop`, and `Postnet_stop`, as applicable to each resource) have executed successfully on all enabled resources in the group. This state also applies before a resource group has started for the first time on the node or zone.
`Online`	The resource group has been started on the node or zone. In other words, the start methods (`Prenet_start`, `Start`, and `Monitor_start`, as applicable to each resource) have executed successfully on all enabled resources in the group.

`Online_faulted`	The resource group was `Pending_online` and has finished starting on this node or zone. However, one or more resources ended up in the `Start_failed` resource state or with `Faulted` status.
`Pending_offline`	The resource group is stopping on the node or zone. The stop methods (`Monitor_stop`, `Stop`, and `Postnet_stop`, as applicable to each resource) are being executed on enabled resources in the group.
`Pending_online`	The resource group is starting on the node or zone. The start methods (`Prenet_start`, `Start`, and `Monitor_start`, as applicable to each resource) are being executed on enabled resources in the group.
`Pending_online_blocked`	The resource group failed to start fully because one or more resources within that resource group have an unsatisfied strong resource dependency on a resource in a different resource group. Such resources remain `Offline`. When the resource dependencies are satisfied, the resource group automatically moves back to the `Pending_online` state.
`Unmanaged`	The initial state of a newly created resource group, or the state of a previously managed resource group. Either `Init` methods have not yet been run on resources in the group, or `Fini` methods have been run on resources in the group. The group is not managed by the RGM.
Default	No default
Tunable	Never

`RG_system` (`boolean`)

If the `RG_system` property is `TRUE` for a resource group, particular operations are restricted for the resource group and for the resources that the resource group contains. This restriction is intended to help prevent accidental modification or deletion of critical

resource groups and resources. Only the clresource(1CL) and clresourcegroup(1CL) commands are affected by this property. Operations for scha_control(1HA) and scha_control(3HA) are not affected.

Before performing a restricted operation on a resource group (or a resource group's resources), you must first set the `RG_system` property of the resource group to `FALSE`. Use care when you modify or delete a resource group that supports cluster services, or when you modify or delete the resources that such a resource group contains.

The following table shows the operations that are restricted for a resource group when `RG_system` is set to `TRUE`.

Operation	Example
Delete a resource group	**`clresourcegroup delete RG1`**
Edit a resource group property (except for `RG_system`)	**`clresourcegroup set -p RG_description=... +`**
Add a resource to a resource group	**`clresource create -g RG1 –t SUNW.nfs R1`** The resource is created in the enabled state and with resource monitoring turned on.
Delete a resource from a resource group	**`clresource delete R1`**
Edit a property of a resource that belongs to a resource group	**`clresource set -g RG1 –t SUNW.nfs -p r_description=″HA-NFS res″ R1`**
Switch a resource group offline	**`clresourcegroup offline RG1`**
Manage a resource group	**`clresourcegroup manage RG1`**
Unmanage a resource group	**`clresourcegroup unmanage RG1`**
Enable a resource	**`clresource enable R1`**
Enable monitoring for a resource	**`clresource monitor R1`**
Disable a resource	**`clresource disable R1`**
Disable monitoring for a resource	**`clresource unmonitor R1`**

If the `RG_system` property is `TRUE` for a resource group, the only property of the resource group that you can edit is the `RG_system` property itself. In other words, editing the `RG_system` property is never restricted.

Default `FALSE`

Tunable Any time

`Suspend_automatic_recovery` (`boolean`)

A Boolean value that indicates whether the automatic recovery of a resource group is suspended. A suspended resource group is *not* automatically restarted or failed over until the cluster administrator explicitly issues the command that resumes automatic recovery. Whether online or offline, suspended data services remain in their current state. You can still manually switch the resource group to a different state on specified nodes or zones. You can also still enable or disable individual resources in the resource group.

If the `Suspend_automatic_recovery` property is set to `TRUE`, automatic recovery of the resource group is suspended. If this property is set to `FALSE`, automatic recovery of the resource group is resumed and active.

The cluster administrator does not set this property directly. The RGM changes the value of the `Suspend_automatic_recovery` property when the cluster administrator suspends or resumes automatic recovery of the resource group. The cluster administrator suspends automatic recovery with the `clresourcegroup suspend` command. The cluster administrator resumes automatic recovery with the `clresourcegroup resume` command. The resource group can be suspended or resumed regardless of the setting of its `RG_system` property.

Default	`FALSE`
Tunable	Never

See Also `projects`(1), clnode(1CL), clresource(1CL), clresourcegroup(1CL), scha_control(1HA), `poolbind`(1M), `zonecfg`(1M), scha_control(3HA), `project`(4), property_attributes(5), r_properties(5), rt_properties(5)

Sun Cluster Concepts Guide for Solaris OS, Chapter 2, "Administering Data Service Resources," in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS* in *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*, *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*

Name r_properties – resource properties

Description The following information describes the resource properties that are defined by Sun Cluster software. These descriptions have been developed for data service developers. For more information about a particular data service, see the man page for that data service.

Note – Scalable, as used in this man page, specifically describes a resource that uses the network load balancing features of Sun Cluster software. Such a resource also uses the properties `Affinity_timeout`, `Generic_affinity`, `Load_balancing_policy`, `Load_balancing_weights`, `Port_list`, `UDP_affinity`, and `Weak_affinity`.

Some resource types can run on multiple nodes or zones without using network load balancing. The `Scalable` property for such a resource is set to `False`, and such a resource does not use the preceding additional properties.

Resource Property Values

Required	The cluster administrator must specify a value when creating a resource with an administrative utility.
Optional	If the cluster administrator does not specify a value when creating a resource group, the system supplies a default value.
Conditional	The Resource Group Manager (RGM) creates the property only if the property is declared in the Resource Type Registration (RTR) file. Otherwise, the property does not exist and is not available to the cluster administrator. A conditional property that is declared in the RTR file is optional or required, depending on whether a default value is specified in the RTR file. For details, see the description of each conditional property.
Query-only	Cannot be set directly by an administrative tool.

The cluster administrator can edit all tunable properties by using the following command:

```
# clresource set -p property=new-value  resource
```

Resource Properties and Descriptions

Note – Property names, such as `Affinity_timeout` and `Cheap_probe_interval`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify property names.

`Affinity_timeout` (`integer`)

Length of time, in seconds, during which connections from a given client IP address for any service in the resource are sent to the same server node or zone. Connections are sent to the same zone provided the scalable service does not use load balancing.

If you set this property to `-1`, all connections are sent to the same node or zone. If you set this property to `0`, all open connections are sent to the same node or zone. If you set this property to *n*, for *n* number of seconds after the last connection has closed, all new connections are sent to the same node or zone as the last connection.

In all cases, if the server node or zone leaves the cluster as a result of a failure, a new server node or zone is selected.

This property is relevant only when `Load_balancing_policy` is either `Lb_sticky` or `Lb_sticky_wild`. In addition, `Weak_affinity` must be set to `False` (the default value).

This property is used only for scalable services.

Category	Conditional/Optional
Default	0
Tunable	Any time

`Cheap_probe_interval` (`integer`)
The number of seconds between invocations of a quick fault probe of the resource. This property is only created by the RGM and available to the cluster administrator if this property is declared in the RTR file.

This property is optional if a default value is specified in the RTR file. If the `Tunable` attribute is not specified in the resource-type file, the `Tunable` value for the property is `When_disabled`.

Category	Conditional
Default	See above
Tunable	When disabled

Extension properties
You declare resource-type properties in the RTR file. The RTR file defines the initial configuration of a data service when the cluster administrator registers the data service with Sun Cluster software.

For information about the individual attributes that you can set for extension properties, see the property_attributes(5) man page.

You can specify that a cluster administrator can set an extension property on a per-node or a per-zone basis or for the entire cluster. However, you cannot specify (in the RTR file) that the cluster administrator can do the same for a system-defined property. System-defined properties implicitly can apply to all nodes or zones or to particular nodes or zones. Whether system-defined properties apply to all or particular nodes or zones depends on the particular definition of the system-defined property.

Category	Conditional
Default	No default
Tunable	Depends on the particular property

`Failover_mode` (`enum`)
Modifies the recovery actions that the RGM takes when a resource fails to start or stop successfully, or when a resource monitor finds a resource to be unhealthy and consequently requests a restart or failover.

`NONE`, `SOFT`, or `HARD` (method failures)
These settings affect only failover behavior when a start or stop method (`Prenet_start`, `Start`, `Monitor_stop`, `Stop`, `Postnet_stop`) fails. The `RESTART_ONLY` and `LOG_ONLY` settings can also affect whether the resource monitor can initiate the execution of the `scha_control`command or the `scha_control()` function.

`NONE` indicates that the RGM is not to take any recovery action when one of the previously mentioned start or stop methods fails. `SOFT` or `HARD` indicates that if a `Start` or `Prenet_start` method fails, the RGM is to relocate the resource's group to a different node or zone. For `Start` or `Prenet_start` failures, `SOFT` and `HARD` are the same.

For failure of a stop method (`Monitor_stop`, `Stop`, or `Postnet_stop`), `SOFT` is the same as `NONE`. If `Failover_mode` is set to `HARD` when one of these stop methods fails, the RGM reboots the node or zone to force the resource group offline. The RGM might then attempt to start the group on another node or zone.

`RESTART_ONLY` or `LOG_ONLY`
Unlike `NONE`, `SOFT`, and `HARD`, which affect failover behavior when a start or stop method fails, `RESTART_ONLY` and `LOG_ONLY` affect all failover behavior. Failover behavior includes monitor-initiated (`scha_control`) restarts of resources and resource groups, and giveovers that are initiated by the resource monitor.

`RESTART_ONLY` indicates that the monitor can run `scha_control` to restart a resource or a resource group. The RGM allows `Retry_count` restarts within `Retry_interval`. If `Retry_count` is exceeded, no further restarts are permitted.

Note – A negative value of `Retry_count`, which is permitted by some but not all resource types, specifies an unlimited number of resource restarts. A more dependable way to specify unlimited restarts is to do the following:

- Set `Retry_interval` to a small value such as 1 or 0.
- Set `Retry_count` to a large value such as 1000.

If the resource type does not declare the `Retry_count` and `Retry_interval` properties, an unlimited number of resource restarts is permitted.

If `Failover_mode` is set to `LOG_ONLY`, no resource restarts or giveovers are permitted. Setting `Failover_mode` to `LOG_ONLY` is the same as setting `Failover_mode` to `RESTART_ONLY` with `Retry_count` set to zero.

`RESTART_ONLY` or `LOG_ONLY` (method failures)
If a `Prenet_start`, `Start`, `Monitor_stop`, `Stop`, or `Postnet_stop` method fails, `RESTART_ONLY` and `LOG_ONLY` are the same as `NONE`. That is, the data service is not failed over or rebooted.

Effect of `Failover_mode` Settings on a Data Service

The effect that each setting for `Failover_mode` has on a data service depends on whether the data service is monitored or unmonitored and whether it is based on the Data Services Development Library (DSDL).

- A data service is monitored if it implements a `Monitor_start` method and monitoring of the resource is enabled. The RGM starts a resource monitor by executing the `Monitor_start` method after starting the resource itself. The resource monitor probes the health of the resource. If the probes fail, the resource monitor might request a restart or a failover by calling the `scha_control()` function. For DSDL-based resources, probes might reveal partial failure (degradation) or a complete failure of the data service. Repeated partial failures accumulate to a complete failure.
- A data service is unmonitored if it does not provide a `Monitor_start` method or if monitoring of the resource has been disabled.
- DSDL-based data services include those that are developed with Agent Builder, through the GDS, or by using the DSDL directly. Some data services, HA Oracle for example, were developed without using the DSDL.

`NONE`, `SOFT`, or `HARD` (probe failures)
If you set `Failover_mode` to `NONE`, `SOFT`, or `HARD`, if the data service is a monitored DSDL-based service, and if the probe fails completely, the monitor calls the `scha_control()` function to request a restart of the resource. If probes continue to fail, the resource is restarted up to a maximum of `Retry_count` times within `Retry_interval`. If the probes fail again after the `Retry_count` number of restarts is reached, the monitor requests a failover of the resource's group to another node or zone.

If you set `Failover_mode` to `NONE`, `SOFT`, or `HARD` and the data service is an unmonitored DSDL-based service, the only failure that is detected is the death of the resource's process tree. If the resource's process tree dies, the resource is restarted.

If the data service is a not a DSDL-based service, the restart or failover behavior depends on how the resource monitor is coded. For example, the Oracle resource monitor recovers by restarting the resource or the resource group, or by failing over the resource group.

`RESTART_ONLY` (probe failures)
If you set `Failover_mode` to `RESTART_ONLY`, if the data service is a monitored DSDL-based service, and if the probe fails completely, the resource is restarted `Retry_count` times within `Retry_interval`. However, if `Retry_count` is exceeded, the resource monitor exits, sets the resource status to `FAULTED`, and generates the status message "Application faulted, but not restarted. Probe quitting." At this point, although monitoring is still enabled, the resource is effectively unmonitored until it is repaired and restarted by the cluster administrator.

If you set `Failover_mode` to `RESTART_ONLY`, if the data service is an unmonitored DSDL-based service, and if the process tree dies, the resource is *not* restarted.

If a monitored data service is not DSDL-based, the recovery behavior depends on how the resource monitor is coded. If you set `Failover_mode` to `RESTART_ONLY`, the resource or resource group can be restarted by a call to the `scha_control()` function `Retry_count` times within `Retry_interval`. If the resource monitor exceeds `Retry_count`, the attempt to restart fails. If the monitor calls `scha_control()` to request a failover, that request fails as well.

`LOG_ONLY` (probe failures)
If you set `Failover_mode` to `LOG_ONLY` for any data service, all `scha_control()` requests to restart the resource or resource group or to fail over the group are precluded. If the data service is DSDL-based, a message is logged when a probe completely fails, but the resource is not restarted. If a probe fails completely more than `Retry_count` times within `Retry_interval`, the resource monitor exits, sets the resource status to `FAULTED`, and generates the status message "Application faulted, but not restarted. Probe quitting." At this point, although monitoring is still enabled, the resource is effectively unmonitored until it is repaired and restarted by the cluster administrator.

If you set `Failover_mode` to `LOG_ONLY`, if the data service is an unmonitored DSDL-based service, and if the process tree dies, a message is logged but the resource is not restarted.

If a monitored data service is not DSDL-based, the recovery behavior depends on how the resource monitor is coded. If you set `Failover_mode` to `LOG_ONLY`, all `scha_control()` requests to restart the resource or resource group or to fail over the group fail.

Category	Optional
Default	`NONE`
Tunable	Any time

`Load_balancing_policy` (`string`)
A string that defines the load-balancing policy in use. This property is used only for scalable services. The RGM automatically creates this property if the `Scalable` property is declared in the RTR file.

`Load_balancing_policy` can take the following values:

- `Lb_weighted` (the default). The load is distributed among various nodes according to the weights set in the `Load_balancing_weights` property.
- `Lb_sticky`. The set of ports is known at the time the application resources are configured. A given client (identified by the client's IP address) of the scalable service is always sent to the same node of the cluster.

- `Lb_sticky_wild`. The port numbers are not known in advance but are dynamically assigned. A given client (identified by the client's IP address) that connects to an IP address of a wildcard sticky service is always sent to the same cluster node regardless of the port number to which that IP address is coming.

Category	Conditional/Optional
Default	`Lb_weighted`
Tunable	At creation

`Load_balancing_weights` (`string_array`)
For scalable resources only. The RGM automatically creates this property if the `Scalable` property is declared in the RTR file. The format is *weight@node,weight@node...*, where *weight* is an integer that reflects the relative portion of load distributed to the specified *node*. The fraction of load distributed to a node is the weight for this node divided by the sum of all weights. For example, `1@1,3@2` specifies that node 1 receives 1/4 of the load and node 2 receives 3/4. The empty string (“”), the default, sets a uniform distribution. Any node that is not assigned an explicit weight receives a default weight of 1. You can specify weight 0 to assign no load to a node.

If the `Tunable` attribute is not specified in the resource-type file, the `Tunable` value for the property is `Anytime`. Changing this property revises the distribution for new connections only.

Category	Conditional/Optional
Default	Null
Tunable	Any time

method`_timeout` for each callback method (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of the method has failed.

Note – You cannot specify a maximum value for a method timeout (using the `Max` attribute). Likewise, you cannot specify a minimum value of zero (`Min=0`).

Category	Conditional/Optional
Default	3,600 (one hour) if the method itself is declared in the RTR file
Tunable	Any time

`Monitored_switch` (`enum`)
You cannot directly set this property. Rather, it is set to `Enabled` or to `Disabled` by the RGM, either on a particular node or zone or for the entire cluster. The RGM does so if the cluster administrator enables or disables the monitor with an administrative utility, either on a particular node or zone or for the entire cluster. If disabled, the `Monitor_start` method is not called on the resource until monitoring is enabled again. If the resource does not have a monitor callback method, this property evaluates to `Disabled`.

Category	Query-only
Default	Enabled, if the resource type has monitoring methods, disabled otherwise
Tunable	See the description

`Network_resources_used` (`string_array`)
A list of logical-hostname or shared-address resources on which this resource has a dependency. This list contains all network-address resources that appear in the properties `Resource_dependencies`, `Resource_dependencies_weak`, `Resource_dependencies_restart`, or `Resource_dependencies_offline_restart`.

The RGM automatically creates this property if the `Scalable` property is declared in the RTR file. If the `Scalable` property is not declared in the RTR file, `Network_resources_used` is unavailable unless it is explicitly declared in the RTR file.

If you do not assign a value to the `Network_resources_used` property, its value is updated automatically by the RGM, based on the setting of the resource-dependencies properties. You do not need to set this property directly. Instead, set the `Resource_dependencies`, `Resource_dependencies_offline_restart`, `Resource_dependencies_restart`, or `Resource_dependencies_weak` property.

To maintain compatibility with earlier releases of Sun Cluster software, you can still set the value of the `Network_resources_used` property directly. If you set the value of the `Network_resources_used` property directly, the value of the `Network_resources_used` property is no longer derived from the settings of the resource-dependencies properties. If you add a resource name to the `Network_resources_used` property, the resource name is automatically added to the `Resource_dependencies` property as well. The only way to remove that dependency is to remove it from the `Network_resources_used` property. If you are not sure whether a network-resource dependency was originally added to the `Resource_dependencies` property or to the `Network_resources_used` property, remove the dependency from both properties. For example, the following command removes a dependency of resource `r1` upon network resource `r2`, regardless of whether the dependency was added to the `Network_resources_used` or `Resource_dependencies` property:

```
# clresource set -p Network_resources_used-=r2 -p Resource_dependencies-=r2 r1
```

Category	Conditional/Optional
Default	The empty list
Tunable	Any time

`Num_resource_restarts` on each cluster node or zone (`integer`)
The number of restart requests that have occurred on this resource within the past *n* seconds, where *n* is the value of the `Retry_interval` property.

A restart request is any of the following calls:

- The `scha_control` command with the `RESOURCE_RESTART` argument
- The `scha_control()` function with the `SCHA_RESOURCE_RESTART` argument
- The `scha_control` command with the `RESOURCE_IS_RESTARTED` argument
- The `scha_control()` function with the `SCHA_RESOURCE_IS_RESTARTED` argument

The RGM resets the restart counter to zero for a given resource on a given node or zone whenever that resource executes one of the following:

- The `scha_control` command with the `GIVEOVER` argument
- The `scha_control()` function with the `SCHA_GIVEOVER` argument

The counter is reset whether the giveover attempt succeeds or fails.

If a resource type does not declare the `Retry_interval` property, the `Num_resource_restarts` property is not available for resources of that type.

Category	Query-only
Default	No default
Tunable	See description

`Num_rg_restarts` on each cluster node or zone (`integer`)
The number of resource-group restart requests that have occurred for this resource within the past *n* seconds, where *n* is the value of the `Retry_interval` property.

A resource-group restart request is either of the following calls:

- The `scha_control` command with the `RESTART` argument
- The `scha_control()` function with the `SCHA_RESTART` argument

If a resource type does not declare the `Retry_interval` property, the `Num_rg_restarts` property is not available for resources of that type.

Category	Query-only
Default	No default
Tunable	See description

`On_off_switch` (`enum`)
You cannot directly set this property. Rather, it is set to `Enabled` or to `Disabled` by the RGM, either on a particular node or zone or for the entire cluster. The RGM does so if the cluster administrator enables or disables the monitor with an administrative utility, either on a particular node or zone or for the entire cluster. If disabled, a resource has no callbacks invoked until it is enabled again.

Category	Query-only
Default	Disabled
Tunable	See description

`Port_list` (`string_array`)
A comma-separated list of port numbers on which the server is listening. Appended to each port number is a slash (`/`) followed by the protocol that is being used by that port, for example, `Port_list=80/tcp` or `Port_list=80/tcp6,40/udp6`.

Possible protocols that you can specify include:

- `tcp`, for only TCP IPv4
- `tcp6`, for both TCP IPv4 and TCP IPv6
- `udp`, for only UDP IPv4
- `udp6`, for both UDP IPv4 and UDP IPv6

If the `Scalable` property is declared in the RTR file, the RGM automatically creates `Port_list`. Otherwise, this property is unavailable unless it is explicitly declared in the RTR file.

Setting up this property for use with Sun Cluster HA for Apache is described in the *Sun Cluster Data Service for Apache Guide for Solaris OS*.

Category	Conditional/Required
Default	No default
Tunable	Any time

`R_description` (`string`)
A brief description of the resource.

Category	Optional
Default	The empty string
Tunable	Any time

`Resource_dependencies` (`string_array`)
A list of resources in the same or in different groups upon which this resource has a strong dependency. This resource cannot be started if the start of any resource in the list fails. If this resource and one of the resources in the list start at the same time, the RGM waits until the resource in the list starts before the RGM starts this resource. If the resource in this resource's `Resource_dependencies` list does not start (for example, if the resource group for the resource in the list remains offline or if the resource in the list is in a `Start_failed` state), this resource also remains offline. If this resource remains offline because of a dependency on a resource in a different resource group that fails to start, this resource's group enters a `Pending_online_blocked` state.

If this resource is brought offline at the same time as those in the list, this resource stops before those in the list. However, if this resource remains online or fails to stop, a resource in the list stops anyway.

By default in a resource group, application resources have an implicit strong resource dependency on network address resources. `Implicit_network_dependencies` in the rg_properties(5) man page contains more information.

Within a resource group, `Prenet_start` methods are run in dependency order before `Start` methods. `Postnet_stop` methods are run in dependency order after `Stop` methods. In different resource groups, the dependent resource waits for the depended-on resource to finish `Prenet_start` and `Start` before it runs `Prenet_start`. The depended-on resource waits for the dependent resource to finish `Stop` and `Postnet_stop` before it runs `Stop`.

To specify the scope of a dependency, append the following qualifiers, including the braces ({ }), to the resource name when you specify this property.

`{LOCAL_NODE}`
: Limits the specified dependency to a per-node basis. The behavior of the dependent is affected by the depended-on resource only on the same node. The dependent resource waits for the depended-on resource to start on the same node. The situation is similar for stopping and restarting.

`{ANY_NODE}`
: Extends the specified dependency to any node. The behavior of the dependent is affected by the depended-on resource on any node. The dependent resource waits for the depended-on resource to start on any primary node before it starts itself. The situation is similar for stopping and restarting.

`{FROM_RG_AFFINITIES}`
: Specifies that the scope of the resource dependency is derived from the `RG_affinities` relationship of the resource groups to which the resources belong. If the dependent resource's group has a positive affinity for the depended-on resource's resource group, and they are starting or stopping on the same node, the dependency is `{LOCAL_NODE}`. If no such positive affinity exists, or if the groups are starting on different nodes, the dependency is `{ANY_NODE}`.

Resource dependencies between two resources that are located in the same resource group are always `{LOCAL_NODE}`.

If you do not specify a qualifier, `FROM_RG_AFFINITIES` is used by default.

Category	Optional
Default	The empty list
Tunable	Any time

`Resource_dependencies_offline_restart` (`string_array`)
: A list of resources in the same or in different groups on which the `Resource_dependencies_offline_restart` resource has an offline-restart dependency.

This property works just as `Resource_dependencies` does, except that, if any resource in the offline-restart dependency list is stopped, this resource is stopped. If that resource in the offline-restart dependency list is subsequently restarted, this resource is restarted.

This resource cannot be started if the start of any resource in the list fails. If this resource and one of the resources in the list start at the same time, the RGM waits until the resource in the list starts before the RGM starts this resource. If the resource in this resource's `Resource_dependencies` list does not start (for example, if the resource group for the resource in the list remains offline or if the resource in the list is in a `Start_failed` state), this resource also remains offline. If this resource remains offline because of a dependency on a resource in a different resource group that fails to start, this resource's group enters a `Pending_online_blocked` state.

If this resource is brought offline at the same time as those in the list, this resource stops before those in the list. However, if this resource remains online or fails to stop, a resource in the list stops anyway.

If a fault occurs on a "depended-on" resource on a node or zone, and the resource cannot recover, the RGM brings that resource on that node or zone offline. The RGM also brings all of the depended-on resource's offline-restart dependents offline by triggering a restart on them. When the cluster administrator resolves the fault and reenables the depended-on resource, the RGM brings the depended-on resource's offline-restart dependents back online as well.

To specify the scope of a dependency, append the following qualifiers, including the braces ({ }), to the resource name when you specify this property.

`{LOCAL_NODE}`
: Limits the specified dependency to a per-node basis. The behavior of the dependent is affected by the depended-on resource only on the same node. The dependent resource waits for the depended-on resource to start on the same node. The situation is similar for stopping and restarting.

`{ANY_NODE}`
: Extends the specified dependency to any node. The behavior of the dependent is affected by the depended-on resource on any node. The dependent resource waits for the depended-on resource to start on any primary node before it starts itself. The situation is similar for stopping and restarting.

`{FROM_RG_AFFINITIES}`
: Specifies that the scope of the resource dependency is derived from the `RG_affinities` relationship of the resource groups to which the resources belong. If the dependent resource's group has a positive affinity for the depended-on resource's resource group, and they are starting or stopping on the same node, the dependency is `{LOCAL_NODE}`. If no such positive affinity exists, or if the groups are starting on different nodes, the dependency is `{ANY_NODE}`.

Resource dependencies between two resources that are located in the same resource group are always `{LOCAL_NODE}`.

If you do not specify a qualifier, `FROM_RG_AFFINITIES` is used by default.

Category	Optional
Default	The empty list
Tunable	Any time

`Resource_dependencies_restart` (`string_array`)
A list of resources in the same or in different groups upon which this resource has an restart dependency.

This property works just as `Resource_dependencies` does, except that, if any resource in the restart dependency list is restarted, this resource is restarted. The restart of this resource occurs after the resource in the list comes back online.

This resource cannot be started if the start of any resource in the list fails. If this resource and one of the resources in the list start at the same time, the RGM waits until the resource in the list starts before the RGM starts this resource.

If the resource in this resource's `Resource_dependencies_restart` list does not start (for example, if the resource group for the resource in the list remains offline or if the resource in the list is in a `Start_failed` state), this resource remains offline. If this resource remains offline because of a dependency on a resource in a different resource group that fails to start, this resource's group enters a `Pending_online_blocked` state.

If this resource is brought offline at the same time as those in the list, this resource stops before those in the list. However, if this resource remains online or fails to stop, a resource in the list stops anyway.

Within a resource group, `Prenet_start` methods are run in dependency order before `Start` methods. `Postnet_stop` methods are run in dependency order after `Stop` methods. In different resource groups, the dependent resource waits for the depended-on resource to finish `Prenet_start` and `Start` before it runs `Prenet_start`. The depended-on resource waits for the dependent resource to finish `Stop` and `Postnet_stop` before it runs `Stop`.

To specify the scope of a dependency, append the following qualifiers, including the braces ({ }), to the resource name when you specify this property.

`{LOCAL_NODE}`
Limits the specified dependency to a per-node basis. The behavior of the dependent is affected by the depended-on resource only on the same node. The dependent resource waits for the depended-on resource to start on the same node. The situation is similar for stopping and restarting.

`{ANY_NODE}`
Extends the specified dependency to any node. The behavior of the dependent is affected by the depended-on resource on any node. The dependent resource waits for the depended-on resource to start on any primary node before it starts itself. The situation is similar for stopping and restarting.

`{FROM_RG_AFFINITIES}`
Specifies that the scope of the resource dependency is derived from the `RG_affinities` relationship of the resource groups to which the resources belong. If the dependent resource's group has a positive affinity for the depended-on resource's resource group, and they are starting or stopping on the same node, the dependency is `{LOCAL_NODE}`. If no such positive affinity exists, or if the groups are starting on different nodes, the dependency is `{ANY_NODE}`.

Resource dependencies between two resources that are located in the same resource group are always `{LOCAL_NODE}`.

If you do not specify a qualifier, `FROM_RG_AFFINITIES` is used by default.

Category	Optional
Default	The empty list
Tunable	Any time

`Resource_dependencies_weak` (`string_array`)
A list of resources in the same or in different groups upon which this resource has a weak dependency. A weak dependency determines the order of method calls within the group. The RGM calls the `Start` methods of the resources in this list before the `Start` method of this resource. The RGM calls the `Stop` methods of this resource before the `Stop` methods of those in the list. The resource can still start if those in the list fail to start or remain offline.

If this resource and a resource in its `Resource_dependencies_weak` list start concurrently, the RGM waits until the resource in the list starts before the RGM starts this resource. If the resource in the list does not start (for example, if the resource group for the resource in the list remains offline or the resource in the list is in a `Start_failed` state), this resource starts. This resource's resource group might enter a `Pending_online_blocked` state temporarily as resources in the this resource's `Resource_dependencies_weak` list start. When all resources in the list have started or failed to start, this resource starts and its group re-enters the `Pending_online` state.

If this resource is brought offline at the same time as those in the list, this resource stops before those in the list. However, if this resource remains online or fails to stop, a resource in the list stops anyway.

Within a resource group, `Prenet_start` methods are run in dependency order before `Start` methods. `Postnet_stop` methods are run in dependency order after `Stop` methods. In different resource groups, the dependent resource waits for the depended-on resource to

finish `Prenet_start` and `Start` before it runs `Prenet_start`. The depended-on resource waits for the dependent resource to finish `Stop` and `Postnet_stop` before it runs `Stop`.

To specify the scope of a dependency, append the following qualifiers, including the braces ({ }), to the resource name when you specify this property.

`{LOCAL_NODE}`
Limits the specified dependency to a per-node basis. The behavior of the dependent is affected by the depended-on resource only on the same node. The dependent resource waits for the depended-on resource to start on the same node. The situation is similar for stopping and restarting.

`{ANY_NODE}`
Extends the specified dependency to any node. The behavior of the dependent is affected by the depended-on resource on any node. The dependent resource waits for the depended-on resource to start on any primary node before it starts itself. The situation is similar for stopping and restarting.

`{FROM_RG_AFFINITIES}`
Specifies that the scope of the resource dependency is derived from the `RG_affinities` relationship of the resource groups to which the resources belong. If the dependent resource's group has a positive affinity for the depended-on resource's resource group, and they are starting or stopping on the same node, the dependency is `{LOCAL_NODE}`. If no such positive affinity exists, or if the groups are starting on different nodes, the dependency is `{ANY_NODE}`.

Resource dependencies between two resources that are located in the same resource group are always `{LOCAL_NODE}`.

If you do not specify a qualifier, `FROM_RG_AFFINITIES` is used by default.

Category	Optional
Default	The empty list
Tunable	Any time

`Resource_name` (`string`)
The name of the resource instance. Must be unique within the cluster configuration and cannot be changed after a resource has been created.

Category	Required
Default	No default
Tunable	Never

`Resource_project_name` (`string`)
The Solaris project name (see `projects`(1)) associated with the resource. Use this property to apply Solaris resource management features such as CPU shares and resource pools to cluster data services. When the RGM brings resources online, it launches the related

processes under this project name. If this property is not specified, the project name will be taken from the `RG_project_name` property of the resource group that contains the resource (see the rg_properties(5) man page). If neither property is specified, the RGM uses the predefined project name `default`. The specified project name must exist in the projects database (see the `projects`(1) man page and *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*).

This property is supported starting with the Solaris 9 release.

Note – Changes to this property take affect the next time the resource is started.

Category	Optional
Default	Null
Tunable	Any time
Valid value	Any valid Solaris project name, or null

`Resource_state` on each cluster node or zone (`enum`)
The RGM-determined state of the resource on each cluster node or zone. Possible states include: `Online`, `Offline`, `Start_failed`, `Stop_failed`, `Monitor_failed`, `Online_not_monitored`, `Starting`, and `Stopping`.

`Online`
The starting methods (`Prenet_start`, `Start`, and `Monitor_start`) have executed successfully on the resource on this node or zone.

`Offline`
The resource has not yet started for the first time on this node or zone, or the stopping methods (`Monitor_stop`, `Stop`, and `Postnet_stop`, as applicable to the particular resource) have executed successfully on the resource on this node or zone.

`Start_failed`
A `Prenet_start` or `Start` method failed on the resource on this node or zone. `Start_failed` means that the method exited with a nonzero exit status or timed out. The service that is represented by the resource might or might not actually have started on this node or zone.

`Stop_failed`
A `Monitor_stop`, `Stop`, or `Postnet_stop` method failed on the resource on this node or zone. `Stop_failed` means that the method exited with a nonzero exit status or timed out. The service that is represented by the resource might or might not actually have stopped on this node or zone.

When a resource enters this state, the resource-group state becomes `Error_stop_failed` and requires you to intervene. `Error_stop_failed` is described in more detail in the rg_properties(5) man page.

`Monitor_failed`
The resource successfully executed its `Prenet_start` or `Start` methods (as applicable to the specific resource type). However, the resources' `Monitor_start` method exited with a nonzero exit status or timed out. The resource monitor might or might not actually have started on this node or zone.

`Online_not_monitored`
The resource successfully executed its `Prenet_start` or `Start` methods (as applicable to the specific resource type). The `Monitor_start` method has not yet been executed on the resource. A resource that is unmonitored (that is, for which there is no `Monitor_start` method, or for which monitoring has been disabled) remains in this state when the resource group goes to `Online` state.

`Starting`
The resource is running the `Prenet_start` or `Start` method in an attempt to go online.

`Stopping`
The resource is running the `Start` or `Postnet_stop` method in an attempt to go offline.

You cannot configure this property.

Category	Query-only
Default	No default
Tunable	Never

`Retry_count` (`integer`)
The number of times a monitor attempts to restart a resource if it fails. If the `Retry_count` is exceeded, depending on the particular data service and the setting of the `Failover_mode` property, the monitor might do one of the following:

- Allow the resource group to remain on the current primary, even though the resource is in a faulted state
- Request a failover of the resource group onto a different node or zone

This property is created by the RGM and is made available to the cluster administrator only if this property is declared in the RTR file. This property is optional if a default value is specified in the RTR file.

If the `Tunable` attribute is not specified in the resource-type file, the `Tunable` value for the property is `When_disabled`.

If you specify a negative value for this property, the monitor attempts to restart the resource an unlimited number of times.

Note – Some resource types do not allow you to set `Retry_count` to a negative value. A more dependable way to specify unlimited restarts is to do the following:

- Set `Retry_interval` to a small value such as 1 or 0.

- Set `Retry_count` to a large value such as 1000.

Category	Conditional
Default	See above
Tunable	When disabled

`Retry_interval` (`integer`)
The number of seconds in which to count attempts to restart a failed resource. The resource monitor uses this property in conjunction with `Retry_count`. This property is created by the RGM and made available to the cluster administrator only if it is declared in the RTR file. This property is optional if a default value is specified in the RTR file.

If the `Tunable` attribute is not specified in the resource-type file, the `Tunable` value for the property is `When_disabled`.

Note – If the `Retry_interval` property is not declared, the call to `scha_resource_get (num_*_restarts)` fails with exit 13 (`SCHA_ERR_RT`).

Category	Conditional
Default	See above
Tunable	When disabled

`Scalable` (`boolean`)
Indicates whether the resource is scalable, that is, whether the resource uses the networking load balancing features of Sun Cluster software.

If this property is declared in the RTR file, the RGM automatically creates the following scalable service properties for resources of that type: `Affinity_timeout`, `Load_balancing_policy`, `Load_balancing_weights`, `Network_resources_used`, `Port_list`, `UDP_affinity`, and `Weak_affinity`. These properties have their default values unless they are explicitly declared in the RTR file. The default for `Scalable`, when it is declared in the RTR file, is `True`.

If this property is declared in the RTR file, it is not permitted to be assigned a `Tunable` attribute other than `At_creation`.

If this property is not declared in the RTR file, the resource is not scalable, you cannot tune this property, and no scalable service properties are set by the RGM. However, you can explicitly declare the `Network_resources_used` and `Port_list` properties in the RTR file, if you want, because these properties can be useful in a non-scalable service as well as in a scalable service.

You use the `Scalable` resource property in combination with the `Failover` resource-type property, as follows:

Failover	Scalable	Description
True	True	Do not specify this illogical combination.
True	False	Specify this combination for a failover service.
False	True	Specify this combination for a scalable service that uses a SharedAddress resource for network load balancing. The *Sun Cluster Concepts Guide for Solaris OS* describes SharedAddress in more detail. You can configure a scalable resource group to run in a non-global zone. But, do not configure a scalable resource to run in multiple non-global zones on the same node.
False	False	Use this combination to configure a multi-master service that does not use network load balancing. You can use a scalable service of this type in zones.

The description for the Failover resource-type property in the rt_properties(5) man page contains additional information.

Category	Optional
Default	See above
Tunable	At creation

Status on each cluster node or zone (enum)
Set by the resource monitor. Possible values are: Online, Degraded, Faulted, Unknown, and Offline. The RGM sets the value to Online when the resource is started, if it is not already set by the Start (or Prenet_start) method. The RGM sets the value to Offline when the resource is stopped, if it is not already set by the Stop (or Postnet_stop) method.

Category	Query-only
Default	No default
Tunable	Only by using the scha_resource_setstatus command

Status_msg on each cluster node or zone (string)
Set by the resource monitor at the same time as the Status property. The RGM sets it to the empty string when the resource is brought Offline, if it was not already set by the Stop (or Postnet_stop) method.

Category	Query-only
Default	No default
Tunable	Only by using the scha_resource_setstatus

`Thorough_probe_interval` (`integer`)
The number of seconds between invocations of a high-overhead fault probe of the resource. This property is created by the RGM and available to the cluster administrator only if it is declared in the RTR file. This property is optional if a default value is specified in the RTR file.

If the `Tunable` attribute is not specified in the resource-type file, the `Tunable` value for the property is `When_disabled`.

Category	Conditional
Default	No default
Tunable	When disabled

`Type` (`string`)
An instance's resource type.

Category	Required
Default	No default
Tunable	Never

`Type_version` (`string`)
Specifies which version of the resource type is currently associated with this resource. The RGM automatically creates this property, which cannot be declared in the RTR file. The value of this property is equal to the `RT_version` property of the resource's type. When a resource is created, the `Type_version` property is not specified explicitly, though it may appear as a suffix of the resource-type name. When a resource is edited, the `Type_version` may be changed to a new value.

Category	See above
Default	None
Tunable	Tunability is derived from the following: ■ The current version of the resource type. ■ The `#$upgrade_from` directive in the resource-type registration file (see the rt_reg(4) man page).

`UDP_affinity` (`boolean`)
If true, all UDP traffic from a given client is sent to the same server node that currently handles all TCP traffic for the client.

This property is relevant only when `Load_balancing_policy` is either `Lb_sticky` or `Lb_sticky_wild`. In addition, `Weak_affinity` must be set to `FALSE` (the default value).

This property is only used for scalable services.

Category	Conditional/Optional

Default	False
Tunable	When disabled

`Weak_affinity` (`boolean`)
If true, enable the weak form of the client affinity. This allows connections from a given client to be sent to the same server node except when a server listener starts (for example, due to a fault monitor restart, a resource failover or switchover, or a node rejoining a cluster after failing) or when `load_balancing_weights` for the scalable resource changes due to an administration action.

Weak affinity provides a low overhead alternative to the default form, both in terms of memory consumption and processor cycles.

This property is relevant only when `Load_balancing_policy` is either `Lb_sticky` or `Lb_sticky_wild`.

This property is only used for scalable services.

Category	Conditional/Optional
Default	False
Tunable	When disabled

See Also `projects`(1), clresource(1CL), clresourcegroup(1CL), clresourcetype(1CL), scha_control(1HA), scha_resource_setstatus(1HA), scha_control(3HA), rt_reg(4), property_attributes(5), rg_properties(5), rt_properties(5)

Sun Cluster Concepts Guide for Solaris OS, *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*

Name rt_properties – resource-type properties

Description The following information describes the resource-type properties that are defined by Sun Cluster software. These descriptions have been developed for data service developers. For information about a particular data service, see the man page for that data service.

Resource-Type Property Values

Required	The property requires an explicit value in the Resource Type Registration (RTR) file. Otherwise, the object to which the property belongs cannot be created. A blank or the empty string is not allowed as a value.
Conditional	To exist, the property must be declared in the RTR file. Otherwise, the RGM does not create the property, and the property is not available to administrative utilities. A blank or the empty string is allowed. If the property is declared in the RTR file but no value is specified, the RGM supplies a default value.
Conditional/Explicit	To exist, the property must be declared in the RTR file with an explicit value. Otherwise, the RGM does not create the property and the property is not available to administrative utilities. A blank or the empty string is not allowed.
Optional	The property can be declared in the RTR file. If the property is not declared in the RTR file, the RGM creates it and supplies a default value. If the property is declared in the RTR file but no value is specified, the RGM supplies the same default value as if the property were not declared in the RTR file.
Query-only	The property cannot be set directly by an administrative utility. The property is not set in the RTR file. The value of the property is provided for information only.

Note – Resource-type properties cannot be updated by administrative utilities with the exception of `Installed_nodes` and `RT_system`. `Installed_nodes` cannot be declared in the RTR file and can only be set by the cluster administrator. `RT_system` can be assigned an initial value in the RTR file, and can also be set by the cluster administrator.

Resource-Type Properties and Descriptions

A resource type is defined by a resource-type registration file that specifies standard and extension property values for the resource type.

Note – resource-type property names, such as `API_version` and `Boot`, are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify property names.

`API_version` (`integer`)
The version of the resource management API that is used by this resource-type implementation.

The following information summarizes the maximum `API_version` that is supported by each release of Sun Cluster software.

Before and up to 3.1	2
3.1 10/03	3
3.1 4/04	4
3.1 9/04	5
3.1 8/05	6
3.2	7
3.2 12/07	8

Declaring a value for `API_version` that is greater than 2 in the RTR file prevents that resource type from being installed on a version of Sun Cluster software that supports a lower maximum version. For example, if you declare `API_version=7` for a resource type, that resource type cannot be installed on any version of Sun Cluster software that was released before the Sun Cluster 3.2 release.

Category	Optional
Default	2
Tunable	Never

`Boot` (`string`)
An optional callback method: the path to the program that the RGM invokes on a node or a zone when the following conditions occur:

- The node or zone joins or rejoins the cluster.
- The resource group that contains the resource of this type is managed.

This method is expected to initialize resources of this type as the `Init` method does.

Category	Conditional/Explicit
Default	None
Tunable	Never

`Failover` (`boolean`)
If you set this property to `TRUE`, resources of this type cannot be configured in any group that can be online on multiple nodes or zones at the same time.

You use this resource-type property in combination with the `Scalable` resource property, as follows:

Failover	Scalable	Description
TRUE	TRUE	Do not specify this illogical combination.
TRUE	FALSE	Specify this combination for a failover service.
FALSE	TRUE	Specify this combination for a scalable service that uses a `SharedAddress` resource for network load balancing. The *Sun Cluster Concepts Guide for Solaris OS* describes `SharedAddress` in more detail. You can configure a scalable resource group to run in a non-global zone. But, do not configure a scalable resource to run in multiple non-global zones on the same node.
FALSE	FALSE	Use this combination to select a multi-master service that does not use network load balancing. You can use a scalable service of this type in zones.

The description for `Scalable` in r_properties(5) and Chapter 3, "Key Concepts for System Administrators and Application Developers," in *Sun Cluster Concepts Guide for Solaris OS* contain additional information.

Category Optional

Default FALSE

Tunable Never

`Fini` (`string`)
An optional callback method: the path to the program that the RGM invokes when a resource of this type is removed from RGM management.

Category Conditional/Explicit

Default No default

Tunable Never

`Global_zone` (`boolean`)
If you set this property to `TRUE` for a resource type, its methods execute in the global zone under all circumstances. If you set this property to `TRUE`, even if the resource group is

running in a non-global zone, the method executes in the global zone. Set this property to `TRUE` only for services that can be managed only from the global zone, such as network addresses and file systems.

Caution – Do not register a resource type for which the `Global_zone` property is set to `TRUE` unless the resource type comes from a known and trusted source. Resource types for which this property is set to `TRUE` circumvent zone isolation and present a risk.

The methods of a resource that is configured to start in a non-global zone and whose `Global_zone` property is set to `TRUE` are always run in the global zone. Such a resource, when configured in a non-global zone, does not benefit from the CPU shares and dedicated processor set configuration. This resource does not benefit even if you set the `RG_slm_type` property to `AUTOMATED`. Sun Cluster software treats such a resource as though it is located in a resource group whose `RG_slm_type` property is set to `MANUAL`.

Because methods for resource types for which the `Global_zone` property is set to `TRUE` run in the global zone, the RGM does not immediately consider these resource types offline when a non-global zone dies. In fact, the RGM runs methods such as `Monitor_stop`, `Stop`, and `Postnet_stop` on these resource types, which include `LogicalHostname`, `SharedAddress`, and `HAStoragePlus`. However, the RGM considers the resources for which the `Global_zone` property is set to `FALSE` to be offline when a non-global zone dies. The RGM cannot run stopping methods on such resources because the methods would have to run in the non-global zone.

Category	Optional
Default	`FALSE`
Tunable	Never

`Init` (`string`)
An optional callback method: the path to the program that the RGM invokes when a resource of this type becomes managed by the RGM.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Init_nodes` (`enum`)
Indicates the nodes or zones on which the RGM is to call the `Init`, `Fini`, `Boot`, and `Validate` methods. You can set this property to `RG_primaries` (just the nodes or zones that can master the resource) or `RT_installed_nodes` (all nodes or zones on which the resource type is installed).

Category	Optional
Default	`RG_primaries`
Tunable	Never

`Installed_nodes` (`string_array`)
A list of the names of nodes or zones on which the resource type is allowed to run. The RGM automatically creates this property. The cluster administrator can set the value. You cannot declare this property in the RTR file.

Category	Can be configured by the cluster administrator
Default	All cluster nodes or zones
Tunable	Any time

`Is_logical_hostname` (`boolean`)
`TRUE` indicates that this resource type is some version of the `LogicalHostname` resource type that manages failover IP addresses.

Category	Query-only
Default	No default
Tunable	Never

`Is_shared_address` (`boolean`)
`TRUE` indicates that this resource type is some version of the `SharedAddress` resource type that manages shared IP (Internet Protocol) addresses.

Category	Query-only
Default	No default
Tunable	Never

`Monitor_check` (`string`)
An optional callback method: the path to the program that the RGM invokes before doing a monitor-requested failover of a resource of this type. If the monitor-check program exits with nonzero on a node or zone, any attempt to fail over to that node or zone is prevented.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Monitor_start` (`string`)
An optional callback method: the path to the program that the RGM invokes to start a fault monitor for a resource of this type.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Monitor_stop` (`string`)
A callback method that is required if `Monitor_start` is set: the path to the program that the RGM invokes to stop a fault monitor for a resource of this type.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Pkglist` (`string_array`)
An optional list of packages that are included in the resource-type installation.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Postnet_stop` (`string`)
An optional callback method: the path to the program that the RGM invokes after calling the `Stop` method of any network-address resources on which a resource of this type depends. This method is expected to perform `Stop` actions that must be performed after network interfaces are configured down.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Prenet_start` (`string`)
An optional callback method: the path to the program that the RGM invokes before calling the `Start` method of any network-address resources on which a resource of this type depends. This method is expected to perform `Start` actions that must be performed before network interfaces are configured up.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Proxy` (`boolean`)
Indicates whether a resource of this type is a proxy resource.

A *proxy resource* is a Sun Cluster resource that imports the state of a resource from another cluster framework such as Oracle Cluster Ready Services (CRS). Oracle CRS, which is now known as Oracle clusterware CRS, is a platform-independent set of system services for cluster environments.

A proxy resource type uses the `Prenet_start` method to start a daemon that monitors the state of the external (proxied) resource. The `Postnet_stop` method stops the monitoring

daemon. The monitoring daemon issues the `scha_control` command with the `CHANGE_STATE_ONLINE` or the `CHANGE_STATE_OFFLINE` tag to set the proxy resource's state to `Online` or to `Offline`, respectively. The `scha_control()` function similarly uses the `SCHA_CHANGE_STATE_ONLINE` and `SCHA_CHANGE_STATE_OFFLINE` tags.

If you set this property to `TRUE`, the resource is a proxy resource.

Category	Optional
Default	`FALSE`
Tunable	Never

`Resource_list` (`string_array`)
The list of all resources of the resource type. The administrator does not set this property directly. Rather, the RGM updates this property when the administrator adds or removes a resource of this type to or from any resource group.

Category	Query-only
Default	Empty list
Tunable	Never

`Resource_type` (`string`)
The name of the resource type. To view the names of the currently registered resource types, type:

`clresourcetype list`

Starting with the Sun Cluster 3.1 release, a resource-type name includes the version, which is mandatory:

`vendor_id.resource_type:version`

The three components of the resource-type name are properties that are specified in the RTR file as *Vendor_id*, *Resource_type*, and *RT_version*. The `clresourcetype` command inserts the period (.) and colon (:) delimiters. The `RT_version` suffix of the resource-type name is the same value as the `RT_version` property. To ensure that the *Vendor_id* is unique, the recommended approach is to use the stock symbol for the company creating the resource type.

Resource-type names that were created before the Sun Cluster 3.1 release continue to use this syntax:

`vendor_id.resource_type`

Category	Required
Default	Empty string
Tunable	Never

`RT_basedir` (`string`)
The directory path that is used to complete relative paths for callback methods. This path is expected to be set to the installation location for the resource-type packages. The path must be a complete path, that is, the path must start with a forward slash (`/`). This property is not required if all the method path names are absolute.

Category	Required, unless all method path names are absolute
Default	No default
Tunable	Never

`RT_description` (`string`)
A brief description of the resource type.

Category	Conditional
Default	Empty string
Tunable	Never

`RT_system` (`boolean`)
If you set this property to `TRUE` for a resource type, you cannot delete the resource type (`clresourcetype unregister` *resource_type_name*). This property is intended to help prevent accidental deletion of resource types, such as `LogicalHostname`, that are used to support the cluster infrastructure. However, you can apply the `RT_system` property to any resource type.

To delete a resource type whose `RT_system` property is set to `TRUE`, you must first set the property to `FALSE`. Use care when you delete a resource type whose resources support cluster services.

Category	Optional
Default	`FALSE`
Tunable	Any time

`RT_version` (`string`)
Starting with the Sun Cluster 3.1 release, a mandatory version string that identifies this resource-type implementation. This property was optional in Sun Cluster 3.0 software. The `RT_version` is the suffix component of the full resource-type name.

Category	Conditional/Explicit or Required
Default	No default
Tunable	Never

`Single_instance` (`boolean`)
If you set this property to `TRUE`, the RGM allows only one resource of this type to exist in the cluster.

Category	Optional
Default	FALSE
Tunable	Never

`Start` (`string`)
A callback method: the path to the program that the RGM invokes to start a resource of this type.

Category	Required, unless the RTR file declares a `Prenet_start` method
Default	No default
Tunable	Never

`Stop` (`string`)
A callback method: the path to the program that the RGM invokes to stop a resource of this type.

Category	Required, unless the RTR file declares a `Postnet_stop` method
Default	No default
Tunable	Never

`Update` (`string`)
An optional callback method: the path to the program that the RGM invokes when properties of a running resource of this type are changed.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Validate` (`string`)
An optional callback method: the path to the program that the RGM invokes to check values for properties of resources of this type.

Category	Conditional/Explicit
Default	No default
Tunable	Never

`Vendor_ID` (`string`)
See the `Resource_type` property.

Category	Conditional
Default	No default
Tunable	Never

See Also clresource(1CL), clresourcegroup(1CL), clresourcetype(1CL), rt_reg(4), SUNW.HAStoragePlus(5), property_attributes(5), r_properties(5), rg_properties(5), scha_control(1HA), scha_control(3HA).

Sun Cluster Concepts Guide for Solaris OS, *Sun Cluster Data Services Developer's Guide for Solaris OS*, *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*

Name scalable_service – scalable resource types

Description A scalable data service is one that takes advantage of the Sun Cluster networking facility. Such a service is implemented as a resource type managed by the Resource Group Manager (RGM).

Standard Resource Properties

The standard resource properties `Scalable`, `Network_resources_used`, `Port_list`, `Load_balancing_policy`, and `Load_balancing_weights` are common to all scalable resource types. See the r_properties(5) man page for the syntax and description of these properties.

Some data services can run in either a scalable or non-scalable mode. These services permit you to specify a value of `True` or `False` for the `Scalable` property at the time that the resource is created. If this property is set to `True` on a resource, the resource is said to be in "scalable mode." The resource then must be contained in a scalable mode resource group, that is, a group that can have its `Maximum_primaries` property set to a value that is greater than 1.

For a data service that can only run in scalable mode, the `Scalable` property is implicitly True for resources of this type, and cannot be changed by the adminstrator.

You can change the `Load_balancing_weights` and `Port_list` properties at any time, even while the resource is online. `Network_resources_used` and `Load_balancing_policy` are set when the resource is created, and you cannot edit these properties afterward. Depending on how the resource type is implemented, these properties might have default values, or you might be required to provide values when you create the resource.

Network Monitoring

A scalable service instance running on a particular node or zone needs to be able to reply to clients over the public networks. The RGM automatically monitors the health of the public networks on nodes or zones where scalable services are to run, and might bring down a scalable service instance on a particular node or zone if the public network becomes inaccessible from that node or zone. If you disable monitoring on a scalable resource by using the `clresource unmonitor` command, these network checks are disabled.

Resource Validatation

When the `Scalable` resource property that is set to `True` is created or updated, the RGM validates various resource properties and will reject the attempted update if these properties are not configured correctly. Among the checks that are performed are the following:

- The `Network_resources_used` property must not be empty. This property must contain the names of existing `SharedAddress` resources. Every node or zone that you specify for the `Nodelist` property of the resource group that contains the scalable resource must appear in either the `NetIfList` property or the `AuxNodeList` property of one of the `SharedAddress` resources.

  Note that you can also specify a zone in a scalable resource's `Nodelist` property. To specify a zone for `Nodelist`, use the following syntax:

  nodename:*zonename*

nodename is the name of the node where *zonename* is located. *zonename* is the name of the zone that you want to include in `Nodelist`. For example, to specify the non-global zone `zone-1` which is located on the node `phys-schost-1`, type:

```
phys-schost-1:zone-1
```

- A resource group that contains a scalable resource must have its `RG_dependencies` property set to include the resource groups of all `SharedAddress` resources listed in the scalable resource's `Network_resources_used` property.
- The `Port_list` property must not be empty. This property must contain a list of port and protocol pairs, where protocol is `tcp`, `tcp6`, `udp`, or `udp6`. Possible protocols that you can specify include `tcp` for only TCP IPv4, `tcp6` for both TCP IPv4 and TCP IPv6, `udp` for only UDP IPv4, or `udp6` for both UDP IPv4 and UDP IPv6.

  For example, you can specify `Port_list=80/tcp,40/udp`.

Affinity IP affinity guarantees that connections from a given client IP address are forwarded to the same cluster node or zone. `Affinity_timeout`, `UDP_affinity`, and `Weak_affinity` are only relevant when `Load_balancing_policy` is set to either `Lb_sticky` or `Lb_sticky_wild`. See r_properties(5) for detailed information.

See Also clresource(1CL), clresourcegroup(1CL), clresourcetype(1CL), rt_callbacks(1HA), rt_reg(4), r_properties(5)

Sun Cluster Software Installation Guide for Solaris OS, *Sun Cluster Data Services Developer's Guide for Solaris OS*

Name SUNW.crs_framework, crs_framework – resource type implementation that coordinates shutdown of Oracle Cluster Ready Services (CRS)

Description The `SUNW.crs_framework` resource type coordinates the shutdown of Oracle Cluster Ready Services (CRS) and Sun Cluster resources in a configuration of Sun Cluster Support for Oracle Real Application Clusters (RAC). This resource type enables Sun Cluster and Oracle CRS to interoperate by enabling Sun Cluster to stop Oracle CRS.

Note – This resource type does *not* enable Oracle CRS to be started by using Sun Cluster administration commands. Oracle CRS can be started only by using Oracle commands or by booting a node.

The CRS voting disk and Oracle cluster registry (OCR) files might reside on storage that is represented by resources of type `SUNW.ScalDeviceGroup` or `SUNW.ScalMountPoint`. In this situation, Oracle CRS must be stopped before resources of type `SUNW.ScalDeviceGroup` or `SUNW.ScalMountPoint` are brought offline. A resource of type `SUNW.crs_framework` ensures that this requirement is met by stopping Oracle CRS processes on a node in the following situations:

- When a resource of type `SUNW.ScalDeviceGroup` or `SUNW.ScalMountPoint` is brought offline on the node by. The CRS processes must be stopped for the following reasons:
  - To ensure that the resource of type `SUNW.ScalDeviceGroup` or `SUNW.ScalMountPoint` is stopped correctly
  - To prevent failure of the database or a node if a resource of type `SUNW.ScalDeviceGroup` or `SUNW.ScalMountPoint` is brought offline while CRS or RAC processes are accessing storage
- When a node is shut down. If the CRS processes are not stopped, the node fails to shut down.

The `SUNW.crs_framework` resource type is a single instance resource type. Only one resource of this type can be created in the cluster.

To ensure that Sun Cluster stops resources in the correct order, configure a resource of type `SUNW.crs_framework` as follows:

- Ensure that any resource group that contains a resource of type `SUNW.ScalDeviceGroup` or `SUNW.ScalMountPoint` declares strong positive affinity for the resource group that is to contain the `SUNW.crs_framework` resource.
- Set an offline-restart dependency by the `SUNW.crs_framework` resource on any resources that represent storage for the CRS voting disk and OCR files. These resources are of type `SUNW.ScalDeviceGroup` or `SUNW.ScalMountPoint`. Limit the scope of each dependency to only the node where the `SUNW.ScalDeviceGroup` resource or `SUNW.ScalMountPoint` resource is running.

- Set a strong dependency by the resource of type `SUNW.crs_framework` on a resource of type `SUNW.rac_framework`.

Create these dependencies and affinities when you configure database resources for the Sun Cluster Support for Oracle RAC data service. For more information, see "Configuring Resources for Oracle RAC Database Instances" in *Sun Cluster Data Service for Oracle RAC Guide for Solaris OS*.

To register this resource type and create instances of this resource type, use one of the following means:

- Sun Cluster Manager
- The clsetup(1CL) utility, specifying the option for configuring Sun Cluster Support for Oracle Real Application Clusters
- The following sequence of Sun Cluster maintenance commands:
  1. To register this resource type, use the clresourcetype(1CL) command.
  2. To create instances of this resource type, use the clresource(1CL) command.

Standard Properties

For a description of all standard resource properties, see the r_properties(5) man page.

Standard resource properties are overridden for this resource type as follows:

`Monitor_start_timeout`

Minimum 60

Default 300

`Monitor_stop_timeout`

Minimum 60

Default 300

`Start_timeout`

Minimum 60

Default 300

`Stop_timeout`

Minimum 60

Default 1200

`Update_timeout`

Minimum 60

Default 300

`Validate_timeout`

Minimum 60

Default 300

Extension Properties The `SUNW.crs_framework` resource type has no extension properties.

Examples

EXAMPLE 1 Creating a `SUNW.crs_framework` Resource

This example registers the `SUNW.crs_framework` resource type and creates an instance of the `SUNW.crs_framework` resource type that is named `crs_framework-rs`. The example makes the following assumptions:

- The C shell is used.
- A resource group that is named `crs-framework-rg` exists.
- The following resources exist:
  - A resource of type `SUNW.rac_framework` that is named `rac_framework-rs`, which represents the RAC framework
  - A resource of type `SUNW.ScalDeviceGroup` that is named `db-storage-rs`, which represents the scalable device group that stores the CRS voting disk and OCR files

```
phys-schost-1# clresourcetype register SUNW.crs_framework
phys-schost-1# clresource create -g crs-framework-rg \
-t SUNW.crs_framework \
-p resource_dependencies=rac_framework-rs \
-p resource_dependencies_offline_restart=db-storage-rs\{local_node\} \
crs_framework-rs
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscucm`

See Also

clresource(1CL), clresourcetype(1CL), clsetup(1CL), `attributes`(5), SUNW.rac_framework(5), SUNW.ScalDeviceGroup(5), SUNW.ScalMountPoint(5)

Sun Cluster Data Service for Oracle RAC Guide for Solaris OS

Name SUNW.derby, derby – resource type implementation of the Java DB database

Description `SUNW.derby` is the failover resource type that enables you to use the Java DB database with Sun Cluster. The Java DB database is based on the Derby database. For information about the database, see *http://db.apache.org/derby/*.

The extension properties associated with the `SUNW.derby` resource type are as follows:

`Child_mon_level`(integer)
Provides control over the processes that are monitored through the process monitor facility. This property denotes the level to which the forked children processes are monitored. Omitting this property or setting this property to the default value is the same as omitting the `-C` option for pmfadm(1M) (all children and their descendents are monitored).

Category	Optional
Default	-1
Tunable	At creation

`DB_path`(string)
Specifies the location of the data file for the Java DB database.

The value for `DB_path` is a string specifying PATH. The specified path should be a path controlled by your chosen storage, for example `HAStoragePlus`.

Category	Mandatory
Default	-1
Tunable	At creation

`DB_port`(integer)
Specifies the port for the Java DB database.

Category	Mandatory
Default	1527
Tunable	At creation

`DB_probe_port`(integer)
Specifies the port that Sun Cluster uses to test the health of the server for the Java DB database.

Category	Mandatory
Default	1528
Tunable	At creation

`Monitor_retry_count`(integer)
Controls fault-monitor restarts. The property indicates the number of times that the process monitor facility restarts the fault monitor. The property corresponds to the `-n` option passed to the pmfadm(1M) command. The Resource Group Manager (RGM) counts the number of restarts in a specified time window (see `Monitor_retry_interval`). Note that `Monitor_retry_count` refers to the restarts of the fault monitor itself, not of the `SUNW.derby` resource.

Category	Optional
Default	2
Tunable	Anytime

`Monitor_retry_interval`(integer)
Indicates the time window in minutes during which the RGM counts fault-monitor failures. The property corresponds to the `-t` option passed to the pmfadm(1M) command. If the number of times that the fault monitor fails exceeds the value of the extension property `Monitor_retry_count`, the process monitor facility does not restart the fault monitor.

Category	Optional
Default	2 minutes
Tunable	Anytime

`Probe_timeout`(integer)
Specifies the timeout value, in seconds, for the probe command.

Category	Optional
Default	30 seconds
Tunable	Anytime

See Also pmfadm(1M)

Name SUNW.Event – resource type implementation for the Cluster Reconfiguration Notification Protocol (CRNP)

Description The `SUNW.Event` resource type implementation provides highly available CRNP services on Sun Cluster. This implementation makes the notification daemon (`/usr/cluster/lib/sc/cl_apid`) highly available by managing it as a resource under the Sun Cluster resource group manager (RGM). The resource group that contains the `SUNW.Event` resource must have a network resource configured in the same resource group. Only a single resource of type `SUNW.Event` should exist on a cluster.

You can run the CRNP only in the global zone.

Standard Properties This section describes key standard properties that control the behavior of the implementation. You use the `clresource` command to set these properties on a `SUNW.Event` resource. The r_properties(5) man page describes these resource properties in more detail.

`Network_resources_used` (`string_array`)
A list of logical-hostname or shared-address network resources upon which this resource has a dependency. This list contains all network-address resources that appear in the properties `Resource_dependencies`, `Resource_dependencies_weak`, `Resource_dependencies_restart`, or `Resource_dependencies_offline_restart`.

This property is updated automatically by the RGM, based on the setting of the resource-dependencies properties. You do not set this property directly. Instead, use the `Resource_dependencies` property.

Category	Conditional/Optional
Default	The empty list
Tunable	Any time

`Port_list` (`string_array`)
A comma-separated list of port numbers on which the server is listening. The r_properties(5) man page describes `Port_list` in more detail.

Category	Conditional/Required
Default	No default
Tunable	Any time

`Resource_dependencies` (`string array`)
A list of resources upon which a resource depends. This list includes any logical-hostname or shared-address network resources that are used by a resource. The default value for this property is null. You must specify this property if the application needs to bind to one or more specific addresses. If no network resource dependencies are specified, the application listens on all addresses.

Before you create the event resource, a `LogicalHostname` or `SharedAddress` resource must already be configured.

You can specify one or more resource names. Each network resource can contain one or more logical hostnames. See the clreslogicalhostname(1CL) and clressharedaddress(1CL) man pages for more information.

You can specify an alternate kind of dependency by using the `Resource_dependencies_weak`, `Resource_dependencies_restart`, or `Resource_dependencies_offline_restart` property instead of the `Resource_dependencies` property. For more information, see the r_properties(5) man page.

Category	Optional
Default	The empty list
Tunable	Any time

`Retry_count` (`integer`)
The number of times that a monitor attempts to restart a resource if it fails. The r_properties(5) man page describes `Retry_count` in more detail.

Note – If you specify a negative value for this property, the monitor attempts to restart the resource an unlimited number of times.

Category	Conditional
Default	2
Tunable	Anytime

`Retry_interval` (`integer`)
The number of seconds over which to count attempts to restart a failed resource. The r_properties(5) man page describes `Retry_interval` in more detail.

Category	Conditional
Default	300
Tunable	Anytime

`Thorough_probe_interval` (`integer`)
The number of seconds between invocations of a high overhead fault probe of the resource. The r_properties(5) man page describes `Thorough_probe_interval` in more detail.

Category	Conditional
Default	60
Tunable	Anytime

Extension Properties

This section describes key extension properties that control the behavior of the implementation.

`Allow_hosts` (`string_array`)

This property controls the set of clients that are allowed to register with the implementation to receive cluster reconfiguration events. The general form of this property is `ipaddress/masklength`, which defines a subnet from which the clients are allowed to register. For example, the setting `129.99.77.0/24` allows clients on the subnet `129.99.77` to register for events. As another example, `192.9.84.231/32` allows only the client `192.9.84.231` to register for events.

In addition, the following special keywords are recognized:

- `LOCAL` refers to all clients that are located in directly connected subnets of the cluster.
- `ALL` allows all clients to register.

Note – If a client matches an entry in both the `Allow_hosts` and the `Deny_hosts` property, that client is prevented from registering with the implementation.

Category	Optional
Default	`LOCAL`
Tunable	Anytime

`Client_retry_count` (`integer`)

This property controls the number of attempts made by the implementation while communicating with external clients. If a client fails to respond within `Client_retry_count` attempts, the client times out. The client is subsequently removed from the list of registered clients that are eligible to receive cluster reconfiguration events. The client must re-register in order to start receiving events again. The section about the `Client_retry_interval` property describes how often these retries are made by the implementation.

Category	Optional
Default	3
Tunable	Anytime

`Client_retry_interval` (`integer`)

This property defines the time period (in seconds) used by the implementation while communicating with unresponsive external clients. Up to `Client_retry_count` attempts are made during this interval to contact the client.

The value for this property can be modified at any time.

Category	Optional
Default	1800
Tunable	Anytime

`Client_timeout` (`integer`)
This property is the timeout value (in seconds) that is used by the implementation while communicating with external clients. However, the implementation continues to attempt to contact the client for a tunable number of times. The sections about the `Client_retry_count` and `Client_retry_interval` properties describe the means of tuning this property.

Category	Optional
Default	60
Tunable	Anytime

`Deny_hosts` (`string_array`)
This property controls the set of clients that are prevented from registering to receive cluster reconfiguration events. To determine access, the settings on this property take precedence over those in the `Allow_hosts` list. The format of this property is the same as the format that is defined in the `Allow_hosts`.

Category	Optional
Default	`NULL`
Tunable	Anytime

`Max_clients` (`integer`)
This property controls the maximum number of clients that can register with the implementation to receive notification of cluster events. Attempts by additional clients to register for events are rejected by the implementation. Since each client registration uses resources on the cluster, tuning this property allows users to control resource usage on the cluster by external clients.

Category	Optional
Default	1000
Tunable	Anytime

Examples

EXAMPLE 1 Creating a `SUNW.Event` Resource With Default Properties

This example shows how to create a failover `SUNW.Event` resource that is named `CRNP` in an existing resource group that is named `events-rg`. The `events-rg` resource group contains a `LogicalHostname` or `SharedAddress` resource, which identifies the failover hostname that is associated with the resource group.

```
# clresourcetype register SUNW.Event
# clresource create -g events-rg -t SUNW.Event CRNP
```

In this example, the `SUNW.Event` resource that is created is named `CRNP`. This resource listens on port 9444 and allows all clients on directly connected subnets to register for events.

EXAMPLE 2 Creating a `SUNW.Event` Resource With Non-Default Properties

This example shows how to create a `SUNW.Event` resource that is named `CRNP` in a resource group that is named `events-rg`. The `CRNP` resource is configured to listen on port 7000, and a specific network resource `foo-1`, which is already configured in the `events-rg` resource group. This CRNP resource allows clients on subnet `192.9.77.0` and clients on directly connected subnets to register, but disallows the client `192.9.77.98` from using the implementation.

```
# clresource create -g events-rg -t SUNW.Event \
-p Port_list=7000/tcp -p Network_resources_used=foo-1 \
-p Allow_hosts=LOCAL,192.9.77.0/24 \
-p Deny_hosts=192.9.77.98/32 CRNP
```

Files

`/usr/cluster/lib/sc/cl_apid`
: CRNP daemon.

`/usr/cluster/lib/sc/events/dtds`
: Directory that contains data type definitions for the CRNP protocol.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu

See Also

clresource(1CL), clresourcegroup(1CL), clresourcetype(1CL), scha_resource_get(1HA), `attributes`(5), r_properties(5)

Name SUNW.gds – resource type for making simple network-aware and nonnetwork-aware applications highly available or scalable

Description The Generic Data Service (GDS) is a mechanism that enables you to make simple network-aware and nonnetwork-aware applications highly available or scalable by plugging them into the Sun Cluster Resource Group Manager (RGM) framework.

The GDS contains a fully functional Sun Cluster resource type, complete with callback methods (rt_callbacks(1HA)) and a Resource Type Registration (RTR) file (rt_reg(4)).

Standard Properties

`Network_resources_used` (`string array`)

A list of logical-hostname or shared-address network resources upon which this resource has a dependency. This list contains all network-address resources that appear in the properties `Resource_dependencies`, `Resource_dependencies_weak`, `Resource_dependencies_restart`, or `Resource_dependencies_offline_restart`.

This property is updated automatically by the RGM, based on the setting of the resource-dependencies properties. You do not set this property directly. Instead, use the `Resource_dependencies` property.

Category	Conditional/Optional
Default	The empty list
Tunable	Any time

`Port_list` (`string array`)

Specifies a comma-separated list of port numbers on which the application listens. See the r_properties(5) man page.

The `Port_list` property must be specified in the start script that Agent Builder creates or with the `clresource` command if you are using Sun Cluster administration commands.

Note – If `Network_aware` is set to `False`, `Port_list` is not required.

Category	Conditional/Required
Default	No default
Tunable	Any time

`Resource_dependencies` (`string array`)

Specifies a list of resources upon which a resource depends. This list includes any logical-hostname or shared-address network resources that are used by a resource. The default value for this property is null. If the `Network_aware` property is set to true, you must set this property to the logical-hostname or shared-address network resources on which the application is listening.

Before you create the GDS resource, a `LogicalHostname` or `SharedAddress` resource must already be configured.

You can specify one or more resource names. Each network resource can contain one or more logical hostnames. See the clreslogicalhostname(1CL) and clressharedaddress(1CL) man pages for more information.

You can specify an alternate kind of dependency by using the `Resource_dependencies_weak`, `Resource_dependencies_restart`, or `Resource_dependencies_offline_restart` property instead of the `Resource_dependencies` property. For more information, see the r_properties(5) man page.

Category	Optional
Default	The empty list
Tunable	Any time

`Start_timeout` (`integer`)
Specifies the timeout value, in seconds, for the start command.

Category	Optional
Default	300 seconds
Tunable	Any time

`Stop_timeout` (`integer`)
Specifies the timeout value, in seconds, for the stop command.

Category	Optional
Default	300 seconds
Tunable	Any time

`Validate_timeout` (`integer`)
Specifies the timeout value, in seconds, for the validate command.

Category	Optional
Default	300 seconds
Tunable	Any time

Extension Properties

`Child_mon_level` (`integer`)
Provides control over the processes that are monitored through the Process Monitor Facility (PMF). This property denotes the level to which the forked children processes are monitored. Omitting this property or setting this property to the default value is the same as omitting the `-C` option for pmfadm(1M): all children (and their descendents) are monitored.

Category	Optional
Default	`-1`

Tunable At creation

`Failover_enabled` (`boolean`)
Allows the resource to fail over. If this property is set to `False`, failover of the resource is disabled. You can use this property to prevent the application resource from initiating a failover of the resource group.

Note – Use the `Failover_mode` property instead of the `Failover_enabled` extension property because `Failover_mode` better controls failover behavior. For more information, see the descriptions of the `LOG_ONLY` and `RESTART_ONLY` values for `Failover_mode` in r_properties(5).

Category Optional

Default `True`

Tunable When disabled

`Log_level` (`enum`)
Specifies the level, or type, of diagnostic messages that are logged by GDS. You can specify `None`, `Info`, or `Err` for this property. When you specify `None`, diagnostic messages are not logged by GDS. When you specify `Info`, both information and error messages are logged. When you specify `Err`, only error messages are logged.

Category Optional

Default `Info`

Tunable Any time

`Network_aware` (`boolean`)
This property specifies whether an application uses the network.

Category Optional

Default `True`

Tunable At creation

`Probe_command` (`string`)
Specifies the command that periodically checks the health of a network-aware or nonnetwork-aware application. This command must be a complete command line that can be passed directly to a shell to probe the application. The probe command returns with an exit status of `0` if the application is running correctly.

The exit status of the probe command is used to determine the severity of the failure of the application. This exit status, called probe status, is an integer between `0` (for success) and `100` (for complete failure). The probe status can also be `201`, which causes the application to fail over unless `Failover_enabled` is set to `False`.

The probe status is used within the GDS probing algorithm to determine whether to restart the application locally or to fail over the application to another node or zone. If the `probe` command is omitted, the GDS provides its own simple probe that connects to the application on the network resource. If the connect succeeds, the GDS disconnects immediately. If both connect and disconnect succeed, the application is deemed to be running correctly.

The GDS does not provide "default" probing behavior for nonnetwork-aware applications. However, a nonnetwork-aware application is started under the PMF, which monitors the application and restarts the application if it fails. The pmfadm(1M) man page contains more information.

Category	Optional
Default	Null
Tunable	When disabled

`Probe_timeout` (`integer`)
Specifies the timeout value, in seconds, for the probe command.

Category	Optional
Default	30 seconds
Tunable	Any time

`Start_command` (`string`)
Specifies the command that starts the application. This command must be a complete command line that can be passed directly to a shell to start the application.

The start command, or one of its forked children, is expected to be a long-running program or daemon which actually provides the service to clients. The start command process tree is monitored by the Process Monitor Facility (PMF) as described under the `Child_mon_level` extension property. If the monitored processes exit, they are restarted according to the settings of the `Retry_count` and `Retry_interval` resource properties. If the retry count is exceeded, an attempt is made to relocate the resource group to a different node or zone.

The exit status that is returned by the start command or its children is ignored.

Category	Required
Default	No default
Tunable	When disabled

`Stop_command` (`string`)
Specifies the command that stops the application. This command must be a complete command line that can be passed directly to a shell to stop the application. If this property is omitted, or if the stop command exits nonzero, the GDS stops the application by using signals.

Category	Optional
Default	Null
Tunable	When disabled

`Stop_signal` (`integer`)
Specifies the signal that stops the application. The values of this property are the same as those defined in `signal`(3HEAD).

Category	Optional
Default	`15`
Tunable	When disabled

`Validate_command` (`string`)
Specifies the absolute path to the command that validates the application. If you do not provide an absolute path, the application is not validated.

The exit status of the validate command is used to determine whether the creation or update of the GDS resource should be permitted. Before creating or updating the resource, the specified validate command is executed on each node of the node list of the resource group that contains the resource. If the validate command exits nonzero, the requested resource creation or update is not permitted. Any output that is written to `stdout` or `stderr` by the validate command will be passed back to the user who issued the administrative command to create or update the resource. Such output can be used to explain why the resource validation failed.

The validate command is also executed when bringing the `gds` resource online, before executing the `Start_command` extension property. If the validate command exits nonzero, it is treated as a start failure.

The validate command is also executed before performing the `GIVEOVER` option of the `scha_control` command to relocate the resource group to a new node or zone. If the command exits nonzero, the giveover is blocked and the resource group remains mastered on its current node or zone.

Category	Optional
Default	Null
Tunable	When disabled

Examples

The following examples show how to use GDS to make an application named `app` highly available. You can also use Sun Cluster Agent Builder (scdsbuilder(1HA)) to create scripts that contain these commands.

Basic Example

This example shows how to register the `SUNW.gds` resource type, create a resource group for the application, create the `LogicalHostname` resource for the logical host name `hhead`, create the network-aware application resource, manage the resource group, enable all its resources, and bring its resources online.

At this point, the application is running, highly available, and monitored by the simple probe that is provided by GDS. You can now check the status of the application.

```
# clresourcetype register SUNW.gds
# clresourcegroup create rg1
# clreslogicalhostname create -g rg1 -h hhead
# clresource create -g rg1 -t SUNW.gds \
-p Start_command="/usr/local/app/bin/start" \
-p Port_list="1234/tcp" -p Network_aware=True \
-p Resource_dependencies=hhead app-rs
# clresourcegroup online -M rg1
# clresourcegroup status +
```

Complex Example

This example shows how to register the `SUNW.gds` resource type, create a resource group for the application, create the `LogicalHostname` resource for the logical host name `hhead`, create the network-aware application resource, log error messages only, manage the resource group, enable all the resources, and bring the resources online.

At this point, the application is running, highly available, and monitored by the fault monitor that is specified by `Probe_command`. You can now check the status of the application.

```
# clresourcetype register SUNW.gds
# clresourcegroup create rg1
# clreslogicalhostname create -g rg1 -h hhead
# clresource create -g rg1 -t SUNW.gds \
-p Start_command="/usr/local/app/bin/start" \
-p Stop_command="/usr/local/app/bin/stop" \
-p Validate_command="/usr/local/app/bin/config" \
-p Probe_command="/usr/local/app/bin/probe" \
-p stop_signal=9 -p failover_enabled=FALSE \
-p Start_timeout=120 -p Stop_timeout=180 \
-p Port_list="1234/tcp" -p Probe_timeout=60 \
-p Network_aware=True \-p Validate_timeout=120 -p Log_level=Err \
-p Resource_dependencies=hhead app-rs
# clresourcegroup online -M rg1
# clresourcegroup status +
```

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscgds`

See Also clreslogicalhostname(1CL), clresource(1CL), clresourcegroup(1CL), clresourcetype(1CL), clressharedaddress(1CL), rt_callbacks(1HA), scdsbuilder(1HA), scha_control(1HA), scha_resource_get(1HA), hatimerun(1M), pmfadm(1M), `signal`(3HEAD), rt_reg(4), `attributes`(5), r_properties(5), scalable_service(5)

Name SUNW.HADerby, HADerby – resource type implementation of the Derby database

Description `SUNW.HADerby` is the failover resource type that enables you to use the Derby data base with Sun Cluster. For information about the Derby database, see *http://db.apache.org/derby/*.

The extension properties associated with the `SUNW.HADerby` resource type are as follows:

`Child_mon_level`(integer)
Provides control over the processes that are monitored through the process monitor facility. This property denotes the level to which the forked children processes are monitored. Omitting this property or setting this property to the default value is the same as omitting the `-C` option for pmfadm(1M) (all children and their descendents are monitored).

Category	Optional
Default	-1
Tunable	At creation

`DB_path`(string)
Specifies the location of the data file for the Derby database.

The value for `DB_path` is a string specifying PATH. The specified path should be a path controlled by your chosen storage, for example `HAStoragePlus`.

Category	Mandatory
Default	-1
Tunable	At creation

`DB_port`(integer)
Specifies the port for the Derby database.

Category	Mandatory
Default	1527
Tunable	At creation

`DB_URL`(string)
Specifies the template for the string of the Java connection to the Derby database.

Category	Mandatory
Default	jdbc:derby://%HOST%:%Port%/SLM
Tunable	At creation

`DB_probe_port`(integer)
Specifies the port that Sun Cluster uses to test the health of the server for the Derby database.

Category	Mandatory
Default	1528
Tunable	At creation

`Monitor_retry_count`(integer)
Controls fault-monitor restarts. The property indicates the number of times that the process monitor facility restarts the fault monitor. The property corresponds to the `-n` option passed to the pmfadm(1M) command. The Resource Group Manager (RGM) counts the number of restarts in a specified time window (see `Monitor_retry_interval`). Note that `Monitor_retry_count` refers to the restarts of the fault monitor itself, not of the `SUNW.HADerby` resource.

Category	Optional
Default	2
Tunable	Anytime

`Monitor_retry_interval`(integer)
Indicates the time window in minutes during which the RGM counts fault-monitor failures. The property corresponds to the `-t` option passed to the pmfadm(1M) command. If the number of times that the fault monitor fails exceeds the value of the extension property `Monitor_retry_count`, the process monitor facility does not restart the fault monitor.

Category	Optional
Default	2 minutes
Tunable	Anytime

`Probe_timeout`(integer)
Specifies the timeout value, in seconds, for the probe command.

Category	Optional
Default	30 seconds
Tunable	Anytime

See Also pmfadm(1M), SUNW.SCTelemetry(5)

Name SUNW.HAStoragePlus – resource type that enforces dependencies between Sun Cluster device services, file systems, and data services

Description `SUNW.HAStoragePlus` describes a resource type that enables you to specify dependencies between data service resources and device groups, cluster file systems, and local file systems.

Note – Local file systems include the UNIX File System (UFS), Quick File System (QFS), Veritas File System (VxFS), and Solaris ZFS (Zettabyte File System).

This resource type enables you to bring data services online only after their dependent device groups and file systems are guaranteed to be available. The `SUNW.HAStoragePlus` resource type also provides support for mounting, unmounting, and checking file systems.

Resource groups by themselves do not provide for direct synchronization with disk device groups, cluster file systems, or local file systems. As a result, during a cluster reboot or failover, an attempt to start a data service can occur while its dependent global devices and file systems are still unavailable. Consequently, the data service's `START` method might time out, and your data service might fail.

The `SUNW.HAStoragePlus` resource type represents the device groups, cluster, and local file systems that are to be used by one or more data service resources. You add a resource of type `SUNW.HAStoragePlus` to a resource group and set up dependencies between other resources and the `SUNW.HAStoragePlus` resource. These dependencies ensure that the data service resources are brought online after the following situations occur:

1. All specified device services are available (and collocated, if necessary).
2. All specified file systems are checked and mounted.

You can also use the `SUNW.HAStoragePlus` resource type to access a local file system from a non-global zone.

Extension Properties The following extension properties are associated with the `SUNW.HAStoragePlus` resource type:

AffinityOn

Specifies whether a `SUNW.HAStoragePlus` resource needs to perform an affinity switchover for all global devices that are defined in the `GlobalDevicePaths` and `FilesystemMountPoints` extension properties. You can specify `TRUE` or `FALSE`. Affinity switchover is set by default, that is, `AffinityOn` is set to `TRUE`.

The `Zpools` extension property ignores the `AffinityOn` extension property. The `AffinityOn` extension property is intended for use with the `GlobalDevicePaths` and `FilesystemMountPoints` extension properties only.

When you set the `AffinityOn` extension property to `FALSE`, the `SUNW.HAStoragePlus` resource passively waits for the specified global services to become available. In this case,

the primary node or zone of each online global device service might not be the same node or zone that is the primary node for the resource group.

The purpose of an affinity switchover is to enhance performance by ensuring the colocation of the device groups and the resource groups on a specific node or zone. Data reads and writes always occur over the device primary paths. Affinity switchovers require the potential primary node list for the resource group and the node list for the device group to be equivalent. The `SUNW.HAStoragePlus` resource performs an affinity switchover for each device service only once, that is, when the `SUNW.HAStoragePlus` resource is brought online.

The setting of the `AffinityOn` flag is ignored for scalable services. Affinity switchovers are not possible with scalable resource groups.

FilesystemCheckCommand

Overrides the check that `SUNW.HAStoragePlus` conducts on each unmounted file system before attempting to mount it. You can specify an alternate command string or executable, which is invoked on all unmounted file systems.

When a `SUNW.HAStoragePlus` resource is configured in a scalable resource group, the file-system check on each unmounted cluster file system is omitted.

The default value for the `FilesystemCheckCommand` extension property is `NULL`. When you set this extension property to `NULL`, Sun Cluster checks UFS or VxFS by issuing the `/usr/sbin/fsck -o p` command. Sun Cluster checks other file systems by issuing the `/usr/sbin/fsck` command. When you set the `FilesystemCheckCommand` extension property to another command string, `SUNW.HAStoragePlus` invokes this command string with the file system mount point as an argument. You can specify any arbitrary executable in this manner. A nonzero return value is treated as an error that occurred during the file system check operation. This error causes the `START` method to fail. When you do not require a file system check operation, set the `FilesystemCheckCommand` extension property to `/bin/true`.

FilesystemMountPoints

Specifies a list of valid file system mount points. You can specify global or local file systems. Global file systems are accessible from all nodes or zones in a cluster. Local file systems are accessible from a single cluster node or zone. Local file systems that are managed by a `SUNW.HAStoragePlus` resource are mounted on a single cluster node or zone. These local file systems require the underlying devices to be Sun Cluster global devices.

These file system mount points are defined in the format `paths[,...]`. You can specify both the path in a non-global zone and the path in a global zone, in this format:

Non-GlobalZonePath:*GlobalZonePath*

The global zone path is optional. If you do not specify a global zone path, Sun Cluster assumes that the path in the non-global zone and in the global zone are the same. If you

specify the path as *Non-GlobalZonePath*:*GlobalZonePath*, you must specify *GlobalZonePath* in the global zone's `/etc/vfstab`.

The default setting for this property is an empty list.

You can use the `SUNW.HAStoragePlus` resource type to make a file system available to a non-global zone. To enable the `SUNW.HAStoragePlus` resource type to do this, you must create a mount point in the global zone and in the non-global zone. The `SUNW.HAStoragePlus` resource type makes the file system available to the non-global zone by mounting the file system in the global zone. The resource type then performs a loopback mount in the non-global zone.

Each file system mount point should have an equivalent entry in `/etc/vfstab` on all cluster nodes and in all global zones. The `SUNW.HAStoragePlus` resource type does not check `/etc/vfstab` in non-global zones.

`SUNW.HAStoragePlus` resources that specify local file systems can only belong in a failover resource group with affinity switchovers enabled. These local file systems can therefore be termed failover file systems. You can specify both local and global file system mount points at the same time.

Any file system whose mount point is present in the `FilesystemMountPoints` extension property is assumed to be local if its `/etc/vfstab` entry satisfies both of the following conditions:

1. The non-global mount option is specified.
2. The "mount at boot" field for the entry is set to "no."

A Solaris ZFS is always a local file system. Do not list a ZFS in `/etc/vfstab`. Also, do not include ZFS mount points in the `FilesystemMountPoints` property.

GlobalDevicePaths

Specifies a list of valid global device group names or global device paths. The paths are defined in the format `paths[,...]`. The default setting for this property is an empty list.

Zpools

Specifies a list of valid ZFS storage pools, each of which contains at least one ZFS. These ZFS storage pools are defined in the format `paths[,...]`. The default setting for this property is an empty list. All file systems in a ZFS storage pool are mounted and unmounted together.

The `Zpools` extension property enables you to specify ZFS storage pools. The devices that make up a ZFS storage pool must be accessible from all the nodes or zones that are configured in the node list of the resource group to which a `SUNW.HAStoragePlus` resource belongs. A `SUNW.HAStoragePlus` resource that manages a ZFS storage pool can only belong to a failover resource group. When a `SUNW.HAStoragePlus` resource that manages a ZFS storage pool is brought online, the ZFS storage pool is imported, and every file system that

the ZFS storage pool contains is mounted. When the resource is taken offline on a node, for each managed ZFS storage pool, all file systems are unmounted and the ZFS storage pool is exported.

Note – `SUNW.HAStoragePlus` does not support file systems created on ZFS volumes.

ZpoolsSearchDir
Specifies the location to search for the devices of `Zpools`. The default value for the `ZpoolsSearchDir` extension property is `/dev/dsk`. The `ZpoolsSearchDir` extension property is similar to the `-d` option of the `zpool` command.

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`SUNWscu`

See Also

rt_reg(4), `attributes`(5)

Warnings

Make data service resources within a given resource group dependent on a `SUNW.HAStoragePlus` resource. Otherwise, no synchronization is possible between the data services and the global devices or file systems. Strong resource dependencies ensure that the `SUNW.HAStoragePlus` resource is brought online before other resources. Local file systems that are managed by a `SUNW.HAStoragePlus` resource are mounted only when the resource is brought online.

Enable logging on UFS systems.

Avoid configuring multiple `SUNW.HAStoragePlus` resources in different resource groups that refer to the same device group and with `AffinityOn` flags set to `TRUE`. Redundant device switchovers can occur. As a result, resource and device groups might be dislocated.

Avoid configuring a ZFS storage pool under multiple `SUNW.HAStoragePlus` resources in different resource groups.

Notes

The `SUNW.HAStoragePlus` resource is capable of mounting any cluster file system that is found in an unmounted state.

All file systems are mounted in the overlay mode.

Local file systems are forcibly unmounted.

The waiting time for all device services and file systems to become available is specified by the `Prenet_Start_Timeout` property in `SUNW.HAStoragePlus`. This is a tunable property.

Name SUNW.Proxy_SMF_failover, Proxy_SMF_failover – resource type for proxying failover SMF services.

Description The `SUNW.Proxy_SMF_failover` resource type represents the proxy for failover of Service Management Facility (SMF) services.

Standard properties and extension properties that are defined for the `SUNW.proxysmffailover` resource type are described in the following subsections. To set these properties for an instance of the `SUNW.Proxy_SMF_failover` resource type, use the `clresource` command (clresource(1CL)).

Standard Properties See r_properties(5) for a description of the following resource properties.

`Start_timeout`
Default: `3600`

Minimum: `60`

`Stop_timeout`
Default: `3600`

Minimum: `60`

`Init_timeout`
Default: `3600`

Minimum: `60`

`Boot_timeout`
Default: `3600`

Minimum: `60`

`Fini_timeout`
Default: `3600`

Minimum: `60`

`Validate_timeout`
Default: `3600`

Minimum: `60`

`Failover_mode`
Default: `SOFT`

Tunable: `Anytime`

`R_description`
Default: `""`

Tunable: `Anytime`

`Retry_count`
Default: `2`

Minimum: `0`

Maximum: `2`

Tunable: `Anytime`

`Retry_interval`
Default: `300`

Tunable: `Anytime`

`Through_probe_interval`
Default: `60`

Tunable: `Anytime`

Extension Properties `Proxied_service_instances`
Includes information about the SMF services to be proxied by the resource. Its value is the path to a file that contains all the proxied SMF services. Each line in the file is dedicated to one SMF service and specifies `svc fmri` and the path to the corresponding service manifest file. For example,if the resource has to manage two services, `restarter_svc_test_1:default` and `restarter_svc_test_2:default`, the file should include the following two lines:

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

Note – The entries above should each appear on a single line. They are broken into multiple lines here for legibility.

Default: `""`

Tunable: `When disabled`

Example This example shows how to register the `SUNW.Proxy_SMF_failover` resource type, create a resource group for the application, create the failover application resource, manage the resource group, and enable a resource.

Register the resource type:

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_failover
```

Create a resource group called `rg1` for the application :

```
# clresourcegroup create rg1
```

Create the failover application resource called `myfailoverres`:

```
# clresource create -t SUNW.Proxy_SMF_failover -g rg1 \\
-x proxied_service_instances="/usr/local/app/svc myfailoverres"
```

where `/usr/local/app/svc` is a text file.

Manage the resource group `rg1`:

```
# clresourcegroup manage rg1
```

Enable the `myfailoverres` resource:

```
# clresource enable myfailoverres
```

Use the following command to check the status of the application:

```
# clresource status
```

See Also pmfadm(1M), scha_resource_get(1HA), clresourcetype(1CL), clresource(1CL), clresourcegroup(1CL), `attributes`(5), r_properties(5)

Sun Cluster Data Services Planning and Administration Guide for Solaris OS

Name SUNW.Proxy_SMF_loadbalanced, Proxy_SMF_loadbalanced – resource type for proxying scalable SMF services

Description The `SUNW.Proxy_SMF_loadbalanced` resource type represents the proxy for scalable Service Management Facility (SMF) services.

Standard properties and extension properties that are defined for the `SUNW.proxysmfloadbalanced` resource type are described in the subsections that follow. To set these properties for an instance of the `SUNW.Proxy_SMF_loadbalanced` resource type, use the `clresource` command (clresource(1CL)).

Standard Properties See r_properties(5) for a description of the following resource properties.

`Start_timeout`
Default: `3600`

Minimum: `60`

`Stop_timeout`
Default: `3600`

Minimum: `60`

`Init_timeout`
Default: `3600`

Minimum: `60`

`Boot_timeout`
Default: `3600`

Minimum: `60`

`Fini_timeout`
Default: `3600`

Minimum: `60`

`Validate_timeout`
Default: `3600`

Minimum: `60`

`Failover_mode`
Default: `SOFT`

Tunable: `Anytime`

`R_description`
Default: `""`

Tunable: `Anytime`

`Retry_count`
Default: `2`

Minimum: `0`

Maximum: `2`

Tunable: `Anytime`

`Retry_interval`
Default: `300`

Tunable: `Anytime`

`Through_probe_interval`
Default: `60`

Tunable: `Anytime`

Extension Properties

`Proxied_service_instances`
Includes information about the SMF services to be proxied by the resource. Its value is the path to a file that contains all the proxied SMF services. Each line in the file is dedicated to one SMF service and specifies `svc fmri` and the path to the corresponding service manifest file. For example,if the resource has to manage two services, `restarter_svc_test_1:default` and `restarter_svc_test_2:default`, the file should include the following two lines:

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

Note – The entries above should each appear on a single line. They are broken into multiple lines here for legibility.

Default: `""`

Tunable: `When disabled`

Example

This example shows how to register the `SUNW.Proxy_SMF_loadbalanced` resource type, create a resource group for the application, create the loadbalanced application resource, manage the resource group, enable all its resources, and bring its resources online.

Register the resource type:

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_loadbalanced
```

Create a resource group called `rg1` for the application :

```
# clresourcegroup create rg1
```

Create the failover application resource called `myloadbalancedres`:

```
# clresource create -t SUNW.Proxy_SMF_loadbalanced -g rg1 \\
-x proxied_service_instances="/usr/local/app/svc myloadbalancedres"
```

where `/usr/local/app/svc` is a text file.

Manage the resource group `rg1`:

```
# clresourcegroup manage rg1
```

Enable the `myloadbalancedres` resource:

```
# clresource enable myloadbalancedres
```

Use the following command to check the status of the application:

```
# clresource status
```

See Also pmfadm(1M), scha_resource_get(1HA), clresourcetype(1CL), clresource(1CL), clresourcegroup(1CL), `attributes`(5), r_properties(5)

Sun Cluster Data Services Planning and Administration Guide for Solaris OS

Name SUNW.Proxy_SMF_multimaster, Proxy_SMF_multimaster – resource type for proxying multi-master SMF services

Description The `SUNW.Proxy_SMF_multimaster` resource type represents the proxy for multi-master Service Management Facility (SMF) services.

Standard properties and extension properties that are defined for the `SUNW.proxysmfmultimaster` resource type are described in the subsections that follow. To set these properties for an instance of the `SUNW.Proxy_SMF_multimaster` resource type, use the `clresource` command (clresource(1CL)).

Standard Properties See r_properties(5) for a description of the following resource properties.

`Start_timeout`
Default: `3600`

Minimum: `60`

`Stop_timeout`
Default: `3600`

Minimum: `60`

`Init_timeout`
Default: `3600`

Minimum: `60`

`Boot_timeout`
Default: `3600`

Minimum: `60`

`Fini_timeout`
Default: `3600`

Minimum: `60`

`Validate_timeout`
Default: `3600`

Minimum: `60`

`Failover_mode`
Default: `SOFT`

Tunable: `Anytime`

`R_description`
Default: `""`

Tunable: `Anytime`

`Retry_count`
Default: `2`

Minimum: `0`

Maximum: `2`

Tunable: `Anytime`

`Retry_interval`
Default: `300`

Tunable: `Anytime`

`Through_probe_interval`
Default: `60`

Tunable: `Anytime`

Extension Properties `Proxied_service_instances`
Includes information about the SMF services to be proxied by the resource. Its value is the path to a file that contains all the proxied SMF services. Each line in the file is dedicated to one SMF service and specifies `svc fmri` and the path to the corresponding service manifest file. For example,if the resource has to manage two services, `restarter_svc_test_1:default` and `restarter_svc_test_2:default`, the file should include the following two lines:

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

Note – The entries above should each appear on a single line. They are broken into multiple lines here for legibility.

Default: `""`

Tunable: `When disabled`

Example This example shows how to register the `SUNW.Proxy_SMF_multimaster` resource type, create a resource group for the application, create the multi-master application resource, manage the resource group, and enable resources.

Register the resource type:

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_multimaster
```

Create a resource group called `rg1` for the application. :

```
# clresourcegroup create rg1
```

Create the failover application resource called `mymultimasterres`:

```
# clresource create -t SUNW.Proxy_SMF_multimaster -g rg1 \\
-x proxied_service_instances="/usr/local/app/svc" mymultimasterres
```

where `/usr/local/app/svc` is a text file.

Manage the resource group `rg1`:

```
# clresourcegroup manage rg1
```

Enable the `mymultimasterres` resource:

```
# clresource enable mymultimasterres
```

Use the following command to check the status of the application:

```
# clresource status
```

See Also pmfadm(1M), scha_resource_get(1HA), clresourcetype(1CL), clresource(1CL), clresourcegroup(1CL), `attributes`(5), r_properties(5)

Sun Cluster Data Services Planning and Administration Guide for Solaris OS

Name SUNW.rac_cvm, rac_cvm – resource type implementation that represents the VERITAS Volume Manager (VxVM) component of Sun Cluster Support for Oracle Real Application Clusters (RAC)

Description The `SUNW.rac_cvm` resource type represents the VxVM component of Sun Cluster Support for Oracle RAC. You can use the `SUNW.rac_cvm` resource type to represent this component *only* if the cluster feature of VxVM is enabled.

Instances of the `SUNW.rac_cvm` resource type hold VxVM component configuration parameters. Instances of this type also show the status of a reconfiguration of the VxVM component.

The `SUNW.rac_cvm` resource type is a single-instance resource type. Only one resource of this type may be created in the cluster.

To register this resource type and create instances of this resource type, use one of the following means:

- Sun Cluster Manager
- The clsetup(1CL) utility, specifying the option for configuring Sun Cluster Support for Oracle Real Application Clusters
- The following sequence of Sun Cluster maintenance commands:
  1. To register this resource type, use the clresourcetype(1CL) command.
  2. To create instances of this resource type, use the clresource(1CL) command.

When a resource of this type is brought offline on a node, the transition from the online state to the offline state requires a finite time to complete. During the transition to the offline state, the resource continues to participate in reconfiguration processes. However, when the resource is offline on a node, changes to resource properties are not effective on the node until the resource is brought back online. Sun Cluster displays a warning message to this effect when a resource of this type is disabled.

The transition to the unmanaged state of a resource group that contains a resource of this type requires a finite time to complete. During the transition to the unmanaged state, the Oracle RAC framework continues to participate in framework reconfiguration processes. However, when the resource group is unmanaged, changes to resource properties are not effective on the node. To stop the Oracle RAC framework, the node must be rebooted.

Many of the extension properties of the `SUNW.rac_cvm` resource type specify timeouts for steps in reconfiguration processes. The optimum values for most of these timeouts are independent of your cluster configuration. Therefore, you should not need to change the timeouts from their default values.

Timeouts that depend on your cluster configuration are described in Guidelines for Setting Timeouts in *Sun Cluster Data Service for Oracle RAC Guide for Solaris OS*. If timeouts occur during reconfiguration processes, increase the values of these timeout properties to accommodate your cluster configuration.

Note – Before you increase the value of a timeout property, consult your VxVM documentation to ensure that the new value is acceptable. The upper limit of the extension property might exceed the maximum value that is acceptable to VxVM.

The extension properties of the `SUNW.rac_cvm` resource type are as follows.

Note – Some extension properties are tunable only when the resource is disabled. You can modify such extension properties only when VxVM is *not* running in cluster mode on any cluster node.

`cvm_abort_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 40. This property specifies the timeout (in seconds) for the abort step of a reconfiguration of the VxVM component of Sun Cluster Support for Oracle RAC. You can modify this property at any time. The modified value is used for the next reconfiguration of the VxVM component.

`cvm_return_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 40. This property specifies the timeout (in seconds) for the return step of a reconfiguration of the VxVM component of Sun Cluster Support for Oracle RAC. You can modify this property at any time. The modified value is used for the next reconfiguration of the VxVM component.

`cvm_start_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for the start step of a reconfiguration of the VxVM component of Sun Cluster Support for Oracle RAC. You can modify this property at any time. The modified value is used for the next reconfiguration of the VxVM component.

`cvm_step1_timeout`
Type integer; minimum 30; maximum 99999; defaults to 100. This property specifies the timeout (in seconds) for step 1 of a reconfiguration of the VxVM component of Sun Cluster Support for Oracle RAC. You can modify this property at any time. The modified value is used for the next reconfiguration of the VxVM component.

`cvm_step2_timeout`
Type integer; minimum 30; maximum 99999; defaults to 100. This property specifies the timeout (in seconds) for step 2 of a reconfiguration of the VxVM component of Sun Cluster Support for Oracle RAC. You can modify this property at any time. The modified value is used for the next reconfiguration of the VxVM component.

`cvm_step3_timeout`
Type integer; minimum 30; maximum 99999; defaults to 240. This property specifies the timeout (in seconds) for step 3 of a reconfiguration of the VxVM component of Sun Cluster Support for Oracle RAC. You can modify this property at any time. The modified value is used for the next reconfiguration of the VxVM component.

`cvm_step4_timeout`
Type integer; minimum 100; maximum 99999; defaults to 320. This property specifies the timeout (in seconds) for step 4 of a reconfiguration of the VxVM component of Sun

Cluster Support for Oracle RAC. You can modify this property at any time. The modified value is used for the next reconfiguration of the VxVM component.

`cvm_stop_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 40. This property specifies the timeout (in seconds) for the stop step of a reconfiguration of the VxVM component of Sun Cluster Support for Oracle RAC. You can modify this property at any time. The modified value is used for the next reconfiguration of the VxVM component.

`vxclust_num_ports`
Type integer; minimum 16; maximum 64; defaults to 32. This property specifies the number of communications ports that the `vxclust` program uses. You can modify this property only when the resource is disabled. The modified value is used for the next reconfiguration of the VxVM component.

`vxclust_port`
Type integer; minimum 1024; maximum 65535; defaults to 5568. This property specifies the communications port number that the `vxclust` program uses. You can modify this property only when the resource is disabled. The modified value is used for the next reconfiguration of the VxVM component.

`vxconfigd_port`
Type integer; minimum 1024; maximum 65535; defaults to 5560. This property specifies the communications port number that the VxVM component configuration daemon `vxconfigd` uses. You can modify this property only when the resource is disabled. The modified value is used for the next reconfiguration of the VxVM component.

`vxkmsgd_port`
Type integer; minimum 1024; maximum 65535; defaults to 5559. This property specifies the communications port number that the VxVM component messaging daemon `vxkmsgd` uses. You can modify this property only when the resource is disabled. The modified value is used for the next reconfiguration of the VxVM component.

Examples

EXAMPLE 1 Creating a `rac_cvm` Resource

This example registers the `SUNW.rac_cvm` resource type and creates an instance of the `SUNW.rac_cvm` resource type that is named `rac_cvm-rs`. The example assumes that the following Sun Cluster objects have been created:

- A resource group that is named `rac-framework-rg`
- A resource of type `SUNW.rac_framework` that is named `rac_framework-rs`

```
phys-schost-1# clresourcetype register SUNW.rac_cvm
phys-schost-1# clresource create -g rac-framework-rg \
-t SUNW.rac_cvm \
-p resource_dependencies=rac_framework-rs rac_cvm-rs
```

EXAMPLE 2 Changing a Property of a `rac_cvm` Resource

This example sets the timeout for step 4 of a reconfiguration of the VxVM component of Sun Cluster Support for Oracle RAC to 300 seconds. The example assumes that an instance of the `SUNW.rac_cvm` resource type named `rac_cvm-rs` has been created.

```
phys-schost-1# clresource set \
-p cvm_step4_timeout=300 rac_cvm-rs
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcvmr

See Also

clresource(1CL), clresourcetype(1CL), clsetup(1CL), `attributes`(5)

Sun Cluster Data Service for Oracle RAC Guide for Solaris OS

Name SUNW.rac_framework, rac_framework – resource type implementation for the framework that enables Sun Cluster Support for Oracle Real Application Clusters (RAC)

Description The `SUNW.rac_framework` resource type represents the framework that enables Sun Cluster Support for Oracle RAC. This resource type enables you to monitor the status of this framework.

The `SUNW.rac_framework` resource type is a single instance resource type. Only one resource of this type may be created in the cluster.

To register this resource type and create instances of this resource type, use one of the following means:

- Sun Cluster Manager
- The clsetup(1CL) utility, specifying the option for configuring Sun Cluster Support for Oracle Real Application Clusters
- The following sequence of Sun Cluster maintenance commands:
  1. To register this resource type, use the clresourcetype(1CL) command.
  2. To create instances of this resource type, use the clresource(1CL) command.

When a resource of this type is brought offline on a node, the transition from the online state to the offline state requires a finite time to complete. During the transition to the offline state, the resource continues to participate in reconfiguration processes. However, when the resource is offline on a node, changes to resource properties are not effective on the node until the resource is brought back online. Sun Cluster displays a warning message to this effect when a resource of this type is disabled.

The transition to the unmanaged state of a resource group that contains a resource of this type requires a finite time to complete. During the transition to the unmanaged state, the Oracle RAC framework continues to participate in framework reconfiguration processes. However, when the resource group is unmanaged, changes to resource properties are not effective on the node. To stop the Oracle RAC framework, the node must be rebooted.

The extension properties of the `SUNW.rac_framework` resource type are as follows:

`reservation_timeout`
Type integer; minimum 100; maximum 99999; defaults to 325. This property specifies the timeout (in seconds) for the reservation step of a reconfiguration of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

Examples **EXAMPLE 1** Creating a `rac_framework` Resource

This example registers the `SUNW.rac_framework` resource type and creates an instance of the `SUNW.rac_framework` resource type named `rac_framework-rs`. The example assumes that a resource group named `rac-framework-rg` has been created.

EXAMPLE 1 Creating a `rac_framework` Resource *(Continued)*

```
phys-host-sc1# clresourcetype register SUNW.rac_framework
phys-host-sc1# clresource create -g rac-framework-rg \
-t SUNW.rac_framework rac_framework-rs
```

EXAMPLE 2 Changing a Property of a `rac_framework` Resource

This example sets the timeout for the reservation step of a reconfiguration of Sun Cluster Support for Oracle RAC to 350 seconds. The example assumes that a resource of type `SUNW.rac_framework` that is named `rac_framework-rs` has been created.

```
phys-host-sc1# clresource set \
-p reservation_timeout=350 rac_framework-rs
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscucm

See Also

clresource(1CL), clresourcetype(1CL), clsetup(1CL), `attributes`(5)

Sun Cluster Data Service for Oracle RAC Guide for Solaris OS

Name SUNW.rac_svm, rac_svm – resource type implementation that represents the Solaris Volume Manager component of Sun Cluster Support for Oracle Real Application Clusters (RAC)

Description The `SUNW.rac_svm` resource type represents the Solaris Volume Manager for Sun Cluster component of the Sun Cluster framework for Oracle RAC.

Instances of the `SUNW.rac_svm` resource type hold Solaris Volume Manager for Sun Cluster component configuration parameters. Instances of this type also show the status of a reconfiguration of the Solaris Volume Manager for Sun Cluster component.

The `SUNW.rac_svm` resource type is a single-instance resource type. Only one resource of this type may be created in the cluster.

To register this resource type and create instances of this resource type, use one of the following means:

- Sun Cluster Manager
- The clsetup(1CL) utility, specifying the option for configuring Sun Cluster Support for Oracle Real Application Clusters
- The following sequence of Sun Cluster maintenance commands:
  1. To register this resource type, use the clresourcetype(1CL) command.
  2. To create instances of this resource type, use the clresource(1CL) command.

When a resource of this type is brought offline on a node, the transition from the online state to the offline state requires a finite time to complete. During the transition to the offline state, the resource continues to participate in reconfiguration processes. However, when the resource is offline on a node, changes to resource properties are not effective on the node until the resource is brought back online. Sun Cluster displays a warning message to this effect when a resource of this type is disabled.

The transition to the unmanaged state of a resource group that contains a resource of this type requires a finite time to complete. During the transition to the unmanaged state, the Oracle RAC framework continues to participate in framework reconfiguration processes. However, when the resource group is unmanaged, changes to resource properties are not effective on the node. To stop the Oracle RAC framework, the node must be rebooted.

The extension properties of the `SUNW.rac_svm` resource type are as follows.

`debug_level`
: Type integer; minimum 0; maximum 10; defaults to 1. This property specifies the debug level for the Solaris Volume Manager for Sun Cluster module of Sun Cluster framework for Oracle RAC. When the debug level is increased, more messages are written to the log files during reconfiguration. You can modify this property at any time.

`svm_abort_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for the abort step of a reconfiguration of the Solaris Volume Manager for Sun Cluster module of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

`svm_return_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for the return step of a reconfiguration of the Solaris Volume Manager for Sun Cluster module of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

`svm_start_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for the start step of a reconfiguration of the Solaris Volume Manager for Sun Cluster module of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

`svm_step1_timeout`
Type integer; minimum 30; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for step 1 of a reconfiguration of the Solaris Volume Manager for Sun Cluster module of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

`svm_step2_timeout`
Type integer; minimum 30; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for step 2 of a reconfiguration of the Solaris Volume Manager for Sun Cluster module of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

`svm_step3_timeout`
Type integer; minimum 30; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for step 3 of a reconfiguration of the Solaris Volume Manager for Sun Cluster module of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

`svm_step4_timeout`
Type integer; minimum 100; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for step 4 of a reconfiguration of the Solaris Volume Manager for Sun Cluster module of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

`svm_stop_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 120. This property specifies the timeout (in seconds) for the stop step of a reconfiguration of the Solaris Volume Manager for Sun Cluster module of Sun Cluster Support for Oracle RAC. You can modify this property at any time.

Examples

EXAMPLE 1 Creating a `rac_svm` Resource

This example registers the `SUNW.rac_svm` resource type and creates an instance of the `SUNW.rac_svm` resource type that is named `rac_svm-rs`. The example assumes that the following Sun Cluster objects have been created:

- A resource group that is named `rac-framework-rg`
- A resource of type `SUNW.rac_framework` that is named `rac_framework-rs`

```
phys-schost-1# clresourcetype register SUNW.rac_svm
phys-schost-1# clresource create -g rac-framework-rg \
-t SUNW.rac_svm \
-p resource_dependencies=rac_framework-rs rac_svm-rs
```

EXAMPLE 2 Changing a Property of a `rac_svm` Resource

This example sets the timeout for step 4 of a reconfiguration of the Solaris Volume Manager for Sun Cluster component of Sun Cluster Support for Oracle RAC to 300 seconds. The example assumes that an instance of the `SUNW.rac_svm` resource type named `rac_svm-rs` has been created.

```
phys-schost-1# clresource set \
-p svm_step4_timeout=300 rac_svm-rs
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscmd

See Also

clresource(1CL), clresourcetype(1CL), clsetup(1CL), `attributes`(5)

Sun Cluster Data Service for Oracle RAC Guide for Solaris OS

Name SUNW.rac_udlm, rac_udlm – resource type implementation for the configuration of the UNIX Distributed Lock Manager (Oracle UDLM) component of Sun Cluster Support for Oracle Real Application Clusters (RAC)

Description The `SUNW.rac_udlm` resource type enables the management of the Oracle UDLM component of Sun Cluster Support for Oracle RAC. The management of this component involves the following activities:

- Setting the parameters of the Oracle UDLM component
- Monitoring the status of the Oracle UDLM component

Note – This resource type is applicable only to the SPARC platform.

The `SUNW.rac_udlm` resource type is a single-instance resource type. Only one resource of this type may be created in the cluster.

To register this resource type and create instances of this resource type, use one of the following means:

- Sun Cluster Manager
- The clsetup(1CL) utility, specifying the option for configuring Sun Cluster Support for Oracle Real Application Clusters
- The following sequence of Sun Cluster maintenance commands:
  1. To register this resource type, use the clresourcetype(1CL) command.
  2. To create instances of this resource type, use the clresource(1CL) command.

When a resource of this type is brought offline on a node, the transition from the online state to the offline state requires a finite time to complete. During the transition to the offline state, the resource continues to participate in reconfiguration processes. However, when the resource is offline on a node, changes to resource properties are not effective on the node until the resource is brought back online. Sun Cluster displays a warning message to this effect when a resource of this type is disabled.

The transition to the unmanaged state of a resource group that contains a resource of this type requires a finite time to complete. During the transition to the unmanaged state, the Oracle RAC framework continues to participate in framework reconfiguration processes. However, when the resource group is unmanaged, changes to resource properties are not effective on the node. To stop the Oracle RAC framework, the node must be rebooted.

The extension properties of the `SUNW.rac_udlm` resource type are as follows.

Note – Some extension properties are tunable only when the resource is disabled. You can modify such extension properties only when the Oracle UDLM is *not* running on any cluster node.

`failfastmode`
Type enum; defaults to `panic`. This property specifies the failfast mode of the node on which the Oracle UDLM is running. The failfast mode determines the action that is performed in response to a critical problem with this node. The possible values of this property are as follows:

`off`	Failfast mode is disabled.
`panic`	The node is forced to panic.

You can modify this property at any time. The modified value is used for the next start-up of the Oracle UDLM. The Oracle UDLM is started when a node is rebooted.

`num_ports`
Type integer; minimum 16; maximum 64; defaults to 32. This property specifies the number of communications ports that the Oracle UDLM uses. You can modify this property only when the resource is disabled. The modified value is used for the next start-up of the Oracle UDLM. The Oracle UDLM is started when a node is rebooted.

`oracle_config_file`
Type string; defaults to `/etc/opt/SUNWcluster/conf/udlm.conf`. This property specifies the configuration file that the Oracle distributed lock manager (DLM) uses. This file must already exist. The file is installed when the Oracle software is installed. For more information, refer to the documentation for the Oracle software. You can modify this property at any time. The modified value is used for the next start-up of the Oracle DLM.

`port`
Type integer; minimum 1024; maximum 65500; defaults to 6000. This property specifies the communications port number that the Oracle UDLM uses. You can modify this property only when the resource is disabled. The modified value is used for the next start-up of the Oracle UDLM. The Oracle UDLM is started when a node is rebooted.

`schedclass`
Type enum; defaults to `RT`. This property specifies the scheduling class of the Oracle UDLM that is passed to the `priocntl`(1) command. The possible values of this property are as follows:

`RT`	Real-time
`TS`	Time-sharing
`IA`	Interactive

You can modify this property only when the resource is disabled. The modified value is used for the next start-up of the Oracle UDLM. The Oracle UDLM is started when a node is rebooted.

`schedpriority`
Type integer; minimum 0; maximum 59; defaults to 11. This property specifies the scheduling priority of the Oracle UDLM that is passed to the `priocntl` command. You can

modify this property only when the resource is disabled. The modified value is used for the next start-up of the Oracle UDLM. The Oracle UDLM is started when a node is rebooted.

`udlm_abort_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 325. This property specifies the timeout (in seconds) for the abort step of an Oracle UDLM reconfiguration. You can modify this property at any time. The modified value is used for the next reconfiguration of the Oracle UDLM.

`udlm_start_step_timeout`
Type integer; minimum 30; maximum 99999; defaults to 100. This property specifies the timeout (in seconds) for the start step of an Oracle UDLM reconfiguration. You can modify this property at any time. The modified value is used for the next start-up of the Oracle UDLM. The Oracle UDLM is started when a node is rebooted.

`udlm_step1_timeout`
Type integer; minimum 30; maximum 99999; defaults to 100. This property specifies the timeout (in seconds) for step 1 of an Oracle UDLM reconfiguration. You can modify this property at any time. The modified value is used for the next reconfiguration of the Oracle UDLM.

`udlm_step2_timeout`
Type integer; minimum 30; maximum 99999; defaults to 100. This property specifies the timeout (in seconds) for step 2 of an Oracle UDLM reconfiguration. You can modify this property at any time. The modified value is used for the next reconfiguration of the Oracle UDLM.

`udlm_step3_timeout`
Type integer; minimum 30; maximum 99999; defaults to 100. This property specifies the timeout (in seconds) for step 3 of an Oracle UDLM reconfiguration. You can modify this property at any time. The modified value is used for the next reconfiguration of the Oracle UDLM.

`udlm_step4_timeout`
Type integer; minimum 30; maximum 99999; defaults to 100. This property specifies the timeout (in seconds) for step 4 of an Oracle UDLM reconfiguration. You can modify this property at any time. The modified value is used for the next reconfiguration of the Oracle UDLM.

`udlm_step5_timeout`
Type integer; minimum 30; maximum 99999; defaults to 100. This property specifies the timeout (in seconds) for step 5 of an Oracle UDLM reconfiguration. You can modify this property at any time. The modified value is used for the next reconfiguration of the Oracle UDLM.

Examples

EXAMPLE 1 Creating a `rac_udlm` Resource

This example registers the `SUNW.rac_udlm` resource type and creates an instance of the `SUNW.rac_udlm` resource type that is named `rac_udlm-rs`. The example assumes that the following Sun Cluster objects have been created:

- A resource group that is named `rac-framework-rg`
- A resource of type `SUNW.rac_framework` that is named `rac_framework-rs`

```
phys-schost-1# clresourcetype register SUNW.rac_udlm
phys-schost-1# clresource create -g rac-framework-rg \
-t SUNW.rac_udlm \
-p resource_dependencies=rac_framework-rs rac_udlm-rs
```

EXAMPLE 2 Changing a Property of a `rac_udlm` Resource

This example sets the timeout for step 4 of a reconfiguration of the Oracle UDLM component of Sun Cluster Support for Oracle RAC to 45 seconds. The example assumes that an instance of the `SUNW.rac_udlm` resource type named `rac_udlm-rs` has been created.

```
phys-schost-1# clresource set \
-p udlm_step4_timeout=45 rac_udlm-rs
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWudlmr

See Also

`priocntl`(1), clresource(1CL), clresourcetype(1CL), clsetup(1CL), `attributes`(5)

Sun Cluster Data Service for Oracle RAC Guide for Solaris OS

Name SUNW.ScalDeviceGroup, ScalDeviceGroup – resource type implementation for a scalable device group

Description The `SUNW.ScalDeviceGroup` resource type represents a scalable device group. An instance of this resource type represents one of the following types of device groups:

- A Solaris Volume Manager for Sun Cluster multi-owner disk set
- A VERITAS Volume Manager with the cluster feature (VxVM) shared-disk group

The `SUNW.ScalDeviceGroup` resource type is a scalable resource type. An instance of this resource type is online on each node in the node list of the resource group that contains the resource.

To register this resource type and create instances of this resource type, use one of the following means:

- Sun Cluster Manager
- The clsetup(1CL) utility, specifying the option for configuring Sun Cluster Support for Oracle Real Application Clusters
- The following sequence of Sun Cluster maintenance commands:
  1. To register this resource type, use the clresourcetype(1CL) command.
  2. To create instances of this resource type, use the clresource(1CL) command.

Standard properties and extension properties that are defined for the `SUNW.ScalDeviceGroup` resource type are described in the subsections that follow.

Standard Properties For a description of all standard resource properties, see the r_properties(5) man page.

Standard resource properties are overridden for this resource type as follows:

`Monitor_Start_timeout`

Minimum 10

Default 300

`Monitor_Stop_timeout`

Minimum 10

Default 300

`Postnet_Stop_timeout`

Minimum 60

Default 300

`Prenet_Start_timeout`

Minimum 60

Default 300

`Start_timeout`

Minimum 60

Default 300

`Stop_timeout`

Minimum 60

Default 300

`Thorough_probe_interval`

Default 300

`Update_timeout`

Minimum 60

Default 300

`Validate_timeout`

Minimum 60

Default 300

Extension Properties The extension properties of this resource type are as follows:

`debug_Level`
This property specifies the level to which debug messages from the resource of this type are logged. When the debug level is increased, more debug messages are written to the log files.

Data type Integer

Default 0

Range 0–10

Tunable Any time

`diskgroupname`
This property specifies the name of the device group that the resource represents. You must set this property to one of the following items:

- The name of an existing Solaris Volume Manager for Sun Cluster multi-owner disk set. This name was specified in the `metaset`(1M) command with which the disk set was created.
- The name of an existing VxVM shared-disk group. This name was specified in the VERITAS command with which the disk group was created.

The requirements for the device group that you specify are as follows:

- The device group must be a valid, existing multi-owner disk set or shared-disk group.
- The device group must be hosted on all nodes that can master the resource.
- The device group must be accessible from all nodes that can master the scalable device group resource.
- The device group must contain at minimum one volume.

Data type	String
Default	No default defined
Range	Not applicable
Tunable	When disabled

`logicaldevicelist`

This property specifies a comma-separated list of logical volumes that the fault monitor of the resource is to monitor. This property is optional. If you do not specify a value for this property, all logical volumes in the device group are to be monitored.

The status of the device group is derived from the statuses of the individual logical volumes that are monitored. If all monitored logical volumes are healthy, the device group is healthy. If any monitored logical volume is faulty, the device group is faulty.

The status of an individual logical volume is obtained by querying the volume's volume manager. If the status of a Solaris Volume Manager for Sun Cluster volume cannot be determined from a query, the fault monitor performs file input/output (I/O) operations to determine the status.

If a device group is discovered to be faulty, monitoring of the resource that represents the group is stopped and the resource is put into the disabled state.

Note – For mirrored disks, if one submirror is faulty, the device group is still considered to be healthy.

The requirements for each logical volume that you specify are as follows:

- The logical volume must exist.
- The logical volume must be contained in the device group that the `diskgroupname` property specifies.
- The logical volume must be accessible from all nodes that can master the scalable device group resource.

Data type	String array
Default	""
Range	Not applicable
Tunable	Any time

`monitor_retry_count`
This property specifies the maximum number of restarts by the process monitor facility (PMF) that are allowed for the fault monitor.

Data type	Integer
Default	4
Range	No range defined
Tunable	Any time

`monitor_retry_interval`
This property specifies the period of time in minutes during which the PMF counts restarts of the fault monitor.

Data type	Integer
Default	2
Range	No range defined
Tunable	Any time

Examples

EXAMPLE 1 Creating a `ScalDeviceGroup` Resource

This example shows the creation of a `ScalDeviceGroup` resource to represent a Solaris Volume Manager for Sun Cluster multi-owner disk set that is named `datadg`. The resource is named `scaldatadg-rs`. This example assumes that the following Sun Cluster objects exist:

- A scalable resource group that is named `scaldatadg-rg`
- An instance of the `SUNW.rac_svm` resource type that is named `rac-svm-rs`

```
# clresourcetype register SUNW.ScalDeviceGroup
# clresource create -t SUNW.ScalDeviceGroup \
-g scaldatadg-rg -p resource_dependencies=rac-svm-rs \
-p diskgroupname=datadg scaldatadg-rs
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu

See Also

clresource(1CL), clresourcetype(1CL), clsetup(1CL), `metaset`(1M), `attributes`(5), r_properties(5), SUNW.rac_cvm(5), SUNW.rac_svm(5)

Sun Cluster Data Service for Oracle RAC Guide for Solaris OS

Notes

In a configuration of Sun Cluster Support for Oracle Real Application Clusters (RAC), a resource of type `SUNW.ScalDeviceGroup` must depend on a resource of the appropriate type.

- If the device group is a Solaris Volume Manager for Sun Cluster multi-owner disk set, the resource must depend on a resource of type `SUNW.rac_svm`. This resource type represents the Solaris Volume Manager for Sun Cluster component of the framework for Sun Cluster Support for Oracle RAC. For more information, see the SUNW.rac_svm(5) man page.
- If the device group is a VxVM shared-disk group, the resource must depend on a resource of type `SUNW.rac_cvm`. This resource type represents the VxVM component of the framework for Sun Cluster Support for Oracle RAC. For more information, see the SUNW.rac_cvm(5) man page.

This dependency ensures that the resource of type `SUNW.ScalDeviceGroup` is brought online only if the resource of type `SUNW.rac_svm` or `SUNW.rac_cvm` is already online.

Create this dependency when you configure storage for the Sun Cluster Support for Oracle RAC data service. For more information, see Chapter 2, "Configuring Storage for Oracle Files," in *Sun Cluster Data Service for Oracle RAC Guide for Solaris OS*in *Sun Cluster Data Service for Oracle RAC Guide for Solaris OS*.

Name SUNW.ScalMountPoint, ScalMountPoint – resource type implementation for a scalable file-system mount point

Description The `SUNW.ScalMountPoint` resource type represents a scalable file-system mount point. An instance of this resource type represents the mount point of one of the following types of file systems:

- A Sun StorEdge QFS shared file system
- A file system on a network-attached storage (NAS) device

  The NAS device and the file system must already be configured for use with Sun Cluster. For more information, see *Sun Cluster 3.1 - 3.2 With Network-Attached Storage Devices Manual for Solaris OS*.

The `SUNW.ScalMountPoint` resource type is a scalable resource type. An instance of this resource type is online on each node in the node list of the resource group that contains the resource.

To register this resource type and create instances of this resource type, use one of the following means:

- Sun Cluster Manager
- The clsetup(1CL) utility, specifying the option for configuring Sun Cluster Support for Oracle Real Application Clusters
- The following sequence of Sun Cluster maintenance commands:
  1. To register this resource type, use the clresourcetype(1CL) command.
  2. To create instances of this resource type, use the clresource(1CL) command.

Standard properties and extension properties that are defined for the `SUNW.ScalMountPoint` resource type are described in the subsections that follow.

Standard Properties For a description of all standard resource properties, see the r_properties(5) man page.

Standard resource properties are overridden for this resource type as follows:

`Monitor_Start_timeout`

Minimum	10
Default	300

`Monitor_Stop_timeout`

Minimum	10
Default	300

`Postnet_Stop_timeout`

Minimum	60

	Default	300
`Prenet_Start_timeout`		
	Minimum	60
	Default	300
`Start_timeout`		
	Minimum	60
	Default	300
`Stop_timeout`		
	Minimum	60
	Default	300
`Thorough_probe_interval`		
	Default	300
`Update_timeout`		
	Minimum	60
	Default	300
`Validate_timeout`		
	Minimum	60
	Default	300

Extension Properties The extension properties of this resource type are as follows:

`debug_Level`
This property specifies the level to which debug messages from the resource for a file-system mount point are logged. When the debug level is increased, more debug messages are written to the log files.

Data type	Integer
Default	0
Range	0–10
Tunable	Any time

`filesystemtype`
This property specifies the type of file system whose mount point the resource represents. You must specify this property. Set this property to one of the following values:

`nas` Specifies that the file system is a file system on a NAS device.

`s-qfs` Specifies that the file system is a Sun StorEdge QFS shared file system.

Data type String

Default No default defined

Range Not applicable

Tunable When disabled

`iotimeout`
This property specifies the timeout value in seconds that the fault monitor uses for file input/output (I/O) probes. To determine if the mounted file system is available, the fault monitor performs I/O operations such as opening, reading, and writing to a test file on the file system. If an I/O operation is not completed within the timeout period, the fault monitor reports an error.

Data type Integer

Default 300

Range 5–300

Tunable Any time

`monitor_retry_count`
This property specifies the maximum number of restarts by the process monitor facility (PMF) that are allowed for the fault monitor.

Data type Integer

Default 4

Range No range defined

Tunable Any time

`monitor_retry_interval`
This property specifies the period of time in minutes during which the PMF counts restarts of the fault monitor.

Data type Integer

Default 2

Range No range defined

Tunable Any time

`mountoptions`
This property specifies a comma-separated list of mount options that are to be used when the file system that the resource represents is mounted. This property is optional. If you do not specify a value for this property, mount options are obtained from the file system's table of defaults.

- For a Sun StorEdge QFS shared file system, these options are obtained from the `/etc/opt/SUNWsamfs/samfs.cmd` file.
- For a file system on a NAS device, these options are obtained from the `/etc/vfstab` file.

Mount options that you specify through this property override the mount options in the file system's table of defaults.

Data type	String
Default	""
Range	Not applicable
Tunable	When disabled

`mountpointdir`
This property specifies the mount point of the file system that the resource represents. The mount point is the full path to the directory where the file system is attached to the file system hierarchy when the file system is mounted. You must specify this property.

The directory that you specify must already exist.

Data type	String
Default	No default defined
Range	Not applicable
Tunable	When disabled

`targetfilesystem`
This property specifies the file system that is to be mounted at the mount point that the `mountpointdir` extension property specifies. You must specify this property. The type of the file system must match the type that the `filesystemtype` property specifies. The format of this property depends on the type of the file system as follows:

- For a Sun StorEdge QFS shared file system, set this property to the name that was assigned to the file system when the file system was created. The file system must be correctly configured. For more information, see your Sun StorEdge QFS shared file system documentation.
- For a file system on a NAS device, set this property to *nas-device*:*path*. The replaceable items in this format are as follows:

  nas-device
  Specifies the name of the NAS device that is exporting the file system. You can optionally qualify this name with a domain.

  path
  Specifies the full path to the file system that the NAS device is exporting.

The NAS device and the file system must already be configured for use with Sun Cluster. For more information, see *Sun Cluster 3.1 - 3.2 With Network-Attached Storage Devices Manual for Solaris OS*.

Data type	String
Default	No default defined
Range	Not applicable
Tunable	When disabled

Examples

EXAMPLE 1 Creating a `ScalMountPoint` Resource

This example shows the creation of a `ScalMountPoint` resource to represent the mount point of a Sun StorEdge QFS shared file system that is used with Solaris Volume manager for Sun Cluster. The resource is named `scal-db_qfs-Data-rs`. The characteristics of the file system are as follows:

- The mount point of the file system is `/db_qfs/Data`.
- The file system that is to be mounted is `Data`.
- Mount options are obtained from the file system's table of defaults, that is the `/etc/opt/SUNWsamfs/samfs.cmd` file.

This example assumes that the following Sun Cluster objects exist:

- A scalable resource group that is named `scaldatadg-rg`
- An instance of the `SUNW.qfs` resource type that is named `qfs-db_qfs-Data-rs`
- An instance of the `SUNW.ScalDeviceGroup` resource type that is named `scaldatadg-rs`

```
# clresourcetype register SUNW.ScalMountPoint
# clresource create -t SUNW.ScalMountPoint \
-g scaldatadg-rg  \
-p resource_dependencies=qfs-db_qfs-Data-rs,scaldatadg-rs \
-p mountpointdir=/db_qfs/Data -p filesystemtype=s-qfs \
-p targetfilesystem=Data scal-db_qfs-Data-rs
```

Attributes

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscu

See Also

clresource(1CL), clresourcetype(1CL), clsetup(1CL), `vfstab`(4), `attributes`(5), r_properties(5), SUNW.ScalDeviceGroup(5)

Sun Cluster Data Service for Oracle RAC Guide for Solaris OS, *Sun Cluster 3.1 - 3.2 With Network-Attached Storage Devices Manual for Solaris OS*

Notes In a configuration of Sun Cluster Support for Oracle Real Application Clusters (RAC), a resource of type `SUNW.ScalMountPoint` that represents the mount point of a Sun StorEdge QFS shared file system must depend on an instance of the `SUNW.qfs` resource type. If you are using Sun StorEdge QFS shared file system with Solaris Volume Manager for Sun Cluster, the resource of `SUNW.ScalMountPoint` type must also depend on an instance of the `SUNW.ScalDeviceGroup` resource type.

These dependencies ensure that the resource of type `SUNW.ScalMountPoint` is brought online only if the resources of type `SUNW.qfs` and `SUNW.DeviceGroup` are already online.

Create these dependencies when you configure storage for the Sun Cluster Support for Oracle RAC data service. For more information, see Chapter 2, "Configuring Storage for Oracle Files," in *Sun Cluster Data Service for Oracle RAC Guide for Solaris OS* in *Sun Cluster Data Service for Oracle RAC Guide for Solaris OS*.

Name SUNW.SCTelemetry, SCTelemetry – resource type for collecting data on system resource usage

Description `SUNW.SCTelemetry` is the resource type that enables you to collect data on the usage of system resources. `SUNW.SCTelemetry` stores system resource usage data in a Java DB database for seven days. The resource of type `SUNW.SCTelemetry` has a dependency on the resource of type `SUNW.derby`. For more information, see the SUNW.derby(5) man page.

The extension properties associated with the `SUNW.SCTelemetry` resource type are as follows:

`Extended_accounting_cleanup`(boolean)
Specifies whether the extended accounting log file is cleaned up, that is whether historical data is deleted. Possible values for `Extended_accounting_cleanup` are `TRUE` and `FALSE`.

Category	Optional
Default	TRUE
Tunable	Anytime

`Monitor_retry_count`(integer)
Controls fault-monitor restarts. The property indicates the number of times that the process monitor facility restarts the fault monitor. The property corresponds to the `-n` option passed to the pmfadm(1M) command. The Resource Group Manager (RGM) counts the number of restarts in a specified time window. See the `Monitor_retry_interval` property for more information. Note that `Monitor_retry_count` refers to the restarts of the fault monitor itself, not to the resource of type `SUNW.SCTelemetry`.

Category	Optional
Default	2
Tunable	Anytime

`Monitor_retry_interval`(integer)
Indicates the time window in minutes during which the RGM counts fault-monitor failures. The property corresponds to the `-t` option passed to the pmfadm(1M) command. If the number of times the fault monitor fails exceeds the value of the `Monitor_retry_count` property, the process monitor facility does not restart the fault monitor.

Category	Optional
Default	2 minutes
Tunable	Anytime

`Probe_timeout`(integer)
Specifies the timeout value, in seconds, for the probe.

Category	Optional

Default	30 seconds
Tunable	Anytime

`Sampling_interval`(integer)
Specifies how often monitoring data is collected. The `Telemetry_sampling_interval` property must have a value of between `30` and `3600`.

Category	Mandatory
Default	60
Tunable	Anytime

See Also pmfadm(1M), SUNW.derby(5)

REFERENCE

SC32 5cl

Name clconfiguration – describes the cluster DTD for the Sun Cluster configuration XML file

Description `clconfiguration` describes the Documentation Type Definition (DTD) for the Sun Cluster configuration eXtensible Markup Language (XML) file. The Sun Cluster configuration file contains Sun Cluster configuration information tagged with XML elements. The file can contain configuration information for one or more clusters, or even for a portion of a cluster. This Sun Cluster configuration information can be used for many cluster functions, including cluster configuration backup and cluster duplication.

The DTD defines the elements, their relationships, and their attributes. The element names reflect the content that they provide. For example, the element `<devicegroup>` defines cluster device groups. Elements might have attributes that are used to modify or refine their properties or characteristics. Many of the object-oriented Sun Cluster commands include an `export` subcommand that exports cluster object information in the format described in the DTD. Many Sun Cluster commands have the option to add, create, and modify Sun Cluster objects through the use of cluster configuration XML data.

Element Hierarchy The following list provides the element hierarchy required by the DTD. This list uses the following defaults for children and attribute properties:

required	Unless otherwise specified, one or more is required.
optional	Unless otherwise specified, there can be zero or one.

```
<propertyList>
  <property>
<state>
<allNodes>

<-- Cluster -->

<cluster>

  <-- Cluster Nodes -->

  <nodelist>
    <node>

  <-- Cluster Transport -->

  <clusterTransport>
    <transportNodeList>
      <transportNode>
        <transportAdapter>
          <transportType>
    <transportSwitchList>
      <transportSwitch>
    <transportCableList>
      <transportCable>
```

```
      <endpoint>

<-- Cluster Global Devices -->

<deviceList>
  <device>
    <devicePath>

<-- Cluster Quorum -->

<clusterQuorum>
  <quorumNodeList>
    <quorumNode>
  <quorumDeviceList>
    <quorumDevice>
      <quorumDevicePathList>
        <quorumDevicePath>

<-- Cluster Device Groups -->

<devicegroupList>
  <devicegroup>
    <memberDeviceList>
      <memberDevice>
    <devicegroupNodeList>
      <devicegroupNode>

<-- Cluster Resource Types -->

<resourcetypeList>
  <resourcetype>
     <resourcetypeRTRFile>
     <resourcetypeNodeList>
       <resourcetypeNode>
       <methodList>
       <method>
       <parameterList>
         <parameter>

<-- Cluster Resources -->

<resourceList>
  <resource>
    <resourceNodeList>
      <resourceNode)
    <monitoredState>
```

```
<-- Cluster Resource Groups -->

<resourcegroupList>
  <resourcegroup>
    <failoverMode>
    <managedState>
    <resourcegroupNodeList>
      <resourcegroupNode>
    <resourcegroupResourceList>
      <resourcegroupResource>

<-- Cluster NAS Devices -->

<nasdeviceList>
  <nasdevice>
     <nasdir>

<-- Cluster SNMP -->

<snmpmibList>
  <snmpmib>
<snmphostList>
  <snmphost>
<snmpuserList>
  <snmpuser>

<-- Cluster Telemetrics -->

<telemetrics>
  <telemetryObjectType>
    <telemetryAttribute>
```

Elements This section lists and describes all of the elements that are defined in the cluster DTD. If an element has required children or attributes, the required default is one. Optional elements default to zero or one.

`<allNodes>`

A list of all member nodes in the cluster. The `<allNodes>` element is a generic element.

The `<allNodes>` element is used to denote all nodes of the cluster.

Parent:	`<resourcetypeNodeList>`
Children:	None
Attributes:	None

`<cluster>`
The root element of a complete cluster configuration XML file. Every cluster configuration XML file must begin with this element as the root. The DTD can accept only one `<cluster>` element. Subsequent `<cluster>` elements in the cluster configuration XML file are ignored.

Parent: None

Children: Optional:

- `<propertyList>`
- `<nodeList>`
- `<clusterTransport>`
- `<deviceList>`
- `<clusterQuorum>`
- `<deviceGroupList>`
- `<resourcetypeList>`
- `<resourcegroupList>`
- `<resourceList>`
- `<nasdeviceList>`
- `<snmpmibList>`
- `<snmphostList>`
- `<snmpuserList>`

Attributes: Required:

- `name`

  The name of the cluster.

`<clusterQuorum>`
The root element of the cluster quorum configuration. All cluster quorum information is defined in the child elements of the `<clusterQuorum>` element.

Parent: `<cluster>`

Children: Optional:

- `<quorumDeviceList>`
- `<quorumNodeList>`

Attributes: None

`<clusterTransport>`
The root element of the cluster transport configuration. All cluster transport information is displayed in a sublevel of the `<clusterTransport>` element.

Parent: `<cluster>`.

Children: Optional:

- `<transportNodeList>`

- <transportSwitchList>
- <transportCableList>

Attributes: None

<device>
A cluster device ID pseudo-driver (DID) device.

Parent: <deviceList>

Children: Optional:

- <devicePath> (zero or more)

Attributes: Required:

- ctd

  The UNIX disk name.

- name

  The instance number of the device.

<devicegroup>
The root element of a cluster device-group instance. All aspects of an individual device group are defined in the child elements of the <devicegroup> element.

Parent: <devicegroupList>

Children: Optional:

- <devicegroupNodeList>
- <memberDeviceList>
- <propertyList>

Attributes: Required:

- name

  The name of the device group. The name attribute can be any valid sequence of characters.

- type

  The type of the device group. The type attribute can have a value of rawdisk, vxvm, svm, or sds.

<devicegroupList>
A list of all the cluster device groups.

Parent: <cluster>

Children: Optional:

- `<devicegroup>`

  One `<devicegroup>` element can be used for each device group in the cluster.

Attributes: None

`<devicegroupNode>`
The node on which a device group is located.

Parent: `<devicegroupNodeList>`

Children: None

Attributes: Required:

- `nodeRef`

  Specifies the name of a cluster node.

`<devicegroupNodeList>`
A list of nodes on which a device group is located.

Parent: `<devicegroup>` (1 or more)

Children: Required:

- `<devicegroupNode> (1 or more)`

Attributes: None

`<deviceList>`
A list of cluster DID devices.

Parent: `<cluster>`

Children: Optional:

- `<device>`

Attributes: Fixed:

- `readonly`

  The `readonly` attribute has a fixed value of `true`.

`<deviceNode>`
The node and disk device on which a particular `<device>` exists.

Parent: `<device>`

Children: None

Attributes: Required:

- `nodeRef`

The name of the node on which an instance exists.

`<endpoint>`

One of the transport endpoints.

Parent: `<transportCable>`

Children: None

Attributes: Required:

- `name`

  The name of the adapter or switch.

- `nodeRef`

  The name of the node that hosts the specified adapter. The `nodeRef` attribute is required only if the `type` attribute is set to `adapter`.

- `type`

  The `type` attribute can be set to either `adapter` or `switch`.

  If the `type` attribute is set to `adapter`, you must specify a `nodeRef` attribute.

  If the `type` attribute is set to `switch`, you can specify a `port` attribute. However, the `port` attribute is not required.

Optional:

- `port`

  The number of the port on the switch. Specify the `port` attribute only if the `type` attribute is set to `switch`.

`<failoverMode>`

The failover mode of a resource group.

Parent: `<resourcegroup>`

Children: None

Attributes: Required:

- `value`

The `value` attribute can be set to `failover` or `scalable`.

`<managedState>`
Indicates whether a resource group is managed or unmanaged.

Parent: `<resourcegroup>`

Children: None

Attributes: Required:

- `value`

  The `value` attribute can be either `true` or `false`.

`<memberDevice>`
The member name of a particular device group. If the `<devicegroup>` is a set of type `rawdisk`, then you must specify one or more `<member>` elements, each with the name of the raw-disk path.

Parent: `<memberDeviceList>`

Children: None

Attributes: Required:

- `name`

  The name of the member.

`<memberDeviceList>`
A list of device group members.

Parent: `<devicegroup>` (one or more)

Children: Required:

- `<memberDevice>`

Attributes: None

`<method>`
Mapping between a generic method type and the actual method name for a specific resource type.

Parent: `<methodList>`

Children: None

Attributes: Required:

- `name`

  The actual name of the method for the resource type.

- `type`

  The type of method for the resource type. You can specify the following types:

  - `MONITOR_CHECK`
  - `MONITOR_START`
  - `MONITOR_STOP`
  - `PRENET_START`
  - `START`
  - `STOP`
  - `VALIDATE`
  - `UPDATE`

`<methodList>`
A list of all of the `<method>` elements that are available for a specific `<resourcetype>`.

Parent: `<resourcetype>`

Children: Optional:

- `<method>`

Attributes: Fixed:

- `readonly`

  The `readonly` attribute has a fixed value of `true`.

`<monitoredState>`
A Boolean value that indicates a portion of an element's state in the cluster. For example, the `<monitoredState>` of a resource specifies whether the resource is monitored, but does not specify whether the resource is available.

Parent: `<resource>`

Children: None

Attributes: Required:

- `value`

  The `value` attribute can be set to `true` or `false`.

`<nasdevice>`
A single instance of a NAS device on the cluster.

Parent: `<nasdeviceList>`

Children: Optional:

- `<nasdir>`

Attributes: Required:

- `name`

  The hostname of the NAS device.

- `type`

  The type of NAS device. You must specify either `sun` or `netapp`.

Optional:

- `userid`

  The user name that is required to access the NAS device.

`<nasdeviceList>`
A list of all NAS devices on the cluster.

Parent:	`<cluster>`
Children:	Optional: - `<nasdevice>`
Attributes:	None

`<nasdir>`
One directory on a NAS device. Each NAS device can have multiple NAS directories.

Parent:	`<nasdevice>`
Children:	None
Attributes:	Required: - `path` The path to the NAS directory.

`<node>`
A cluster node. Specify one `<node>` element for each node in the cluster.

Parent:	`<nodeList>`
Children:	Optional: - `<propertyList>`
Attributes:	Required: - `name` Must be equal to the name of the node.

Optional:

- `globaldevfs`

  The path to the global mount directory.

- `id`

  The cluster node ID. If not specified, the cluster node ID attribute is provided a default value of an empty string.

`<nodeList>`
A list of all nodes in the cluster.

Parent: `<cluster>`

Children: Optional:

- `<node>`

  At least one `node` attribute must be supplied for each node on the cluster.

Attributes: None

`<parameter>`
A set of attributes that describes the `<method>` element timeout values and other parameters for a cluster resource type.

Parent: `<parameterList>`

Children: None

Attributes: Required:

- `extension`

  The `extension` attribute can be set to `true` or `false`.

- `name`

  The name of the parameter.

- `tunability`

  The value of the parameter's tunability. The `tunability` attribute can be set to one of the following values: `atCreation`, `anyTime`, or `whenDisabled`.

- `type`

The type of the parameter. The `type` attribute can be set to one of the following values: `boolean`, `enum`, `int` , `string`, or `stringArray`.

Optional:

- `default`

  The default value of this parameter if a value is not explicitly specified. For example, the default for the `method` element timeout is `START`.

- `description`

  A description of the parameter. If not defined, this attribute defaults to an empty string.

- `enumList`

  An enumerated list of objects. For example, the attribute might be a list of failover modes in order of preference.

- `maxLength`

  The maximum length of a `string` or `stringArray` type parameter.

- `minArrayLength`

  The minimum size of an `stringArray` type parameter.

- `minLength`

  The minimum length of a `string` or `stringArray` type parameter.

`<parameterList>`

A list of `<parameter>` elements that describes a resource type.

Parent:	`<resourcetype>`
Children:	Optional: - `<parameter>`
Attributes:	Fixed: - `readonly` The `readonly` attribute has a fixed value of `true`.

`<property>`
A generic element that describes one property. The property is not specific to any subset of cluster related configuration.

Parent: `<propertyList>`

Children: None

Attributes: Required:

- `name`

  The name of the property.

- `value`

  The value of the property.

Optional:

- `readonly`

  The `readonly` attribute can be set to `true` or `false`. If this value is not specified, the attribute defaults to the value `false`.

- type

  The property type.

`<propertyList>`
A list of `<property>` elements. The `<propertyList>` element is a generic element.

Parents: `<cluster>`, `<deviceGroup>`, `<node>`, `<quorumDevice>`, `<quorumNode>`, `<resource>`, `<resourceNode>`, `<resourcegroup>`, `<resourceType>`, `<transportAdapter>`, `<transportType>`

Children: `Optional:`

- `<property>`

Attributes: Optional:

- `extension`

  This attribute can have one of the following values: `true`, `false`, `mixed`, or `doesNotApply`. If a value is not specified, the `extension` attribute has a default value of `doesNotApply`.

- `readonly`

This attribute can have a value of `true` or `false`. If a value is not specified, the `readonly` attribute has a default value of `false`.

`<quorumDevice>`
An individual cluster quorum device.

Parent: `<quorumDeviceList>`

Children: Optional:

- `<propertyList>`

  The `<quorumDevice>` element can have only one `<propertyList>` child.

- `<quorumDevicePathList>`

  The `<quorumDevice>` element can have only one `<quorumDevicePathList>` child.

Attributes: Required:

- `name`

  The name of the quorum device.

- `type`

  The type of quorum device that is referenced by this element. The `type` attribute can be set to `netapp_nas`, `scsi`, or `quorum_server`.

`<quorumDeviceList>`
A list of all quorum devices in the cluster.

Parent: `<clusterQuorum>`

Children: Optional:

- `<quorumDevice>`

Attributes: None

`<quorumDevicePath>`
The path to a cluster quorum device.

Parent: `<quorumDevicePathList>`

Children: Optional:

- `<state>`

The <quorumDevicePath> element can have only one <state> child.

Attributes: Required:

- nodeRef

  The name of the node that the quorum device resides on.

<quorumDevicePathList>
A list of all paths to a particular <quorumDevice>.

Parent: <quorumDevice>

Children: Required:

- <quorumDevicePath>

Attributes: Fixed:

- readonly

  The readonly attribute is set to true.

<quorumNode>
A node in the cluster that participates in the cluster quorum.

Parent: <quorumNodeList>

Children: Optional:

- <propertyList>

Attributes: Required:

- <nodeRef>

  The name of the node.

<quorumNodeList>
A list of all nodes that participate in the cluster quorum. In a functional cluster that is not in installmode, this list typically contains all nodes in the cluster. In a cluster that is still in installmode, this list might contain only one of the cluster nodes.

Parent: <clusterQuorum>

Children: Required:

- <quorumNode>

Attributes: Fixed:

- readonly

  The readonly attribute is set to true.

`<resource>`
A cluster resource.

Parent: `<resourceList>`

Children: Optional:

- `<resourceNodeList>`
- `<propertyList>`

Attributes: Required:

- `name`

  The name of the resource.

- `resourcegroupRef`

  The resource group to which the resource belongs.

- `resourcetypeRef`

  The type of resource that is described by this element.

`<resourceList>`
A list of the root node for the cluster resources that are defined in the configuration.

Parent: `<cluster>`

Children: Optional:

- `<resource>`

Attributes: None

`<resourcegroup>`
A cluster resource group.

Parent: `<resourcegroupList>`

Children: Required:

- `<failoverMode>`
- `<managedState>`
- `<resourcegroupNodeList>`
- `<resourcegroupResourceList>`
- `<propertyList>`

Attributes: Required:

- `name`

  The name of the resource.

`<resourcegroupList>`
The root node for the cluster resource groups that are defined in the configuration.

Parent: `<cluster>`

Children: Optional:

- `<resourcegroup>`

Attributes: None

`<resourcegroupNode>`
The node on which a resource group is defined.

Parent: `<resourcegroupNodeList>`

Children: None

Attributes: Required:

- `nodeRef`

  The name of the cluster node.

Optional:

- Zone

  The name of the zone.

`<resourcegroupNodeList>`
The cluster nodes on which a particular resource group operates.

Parent: `<resourcegroup>`

Children: Required:

- `<resourcegroupNode>`

Attributes: None

`<resourcegroupResource>`
A cluster resource that belongs to a particular resource group.

Parent: `<resourcegroupResourceList>`

Children: None

Attributes: Required:

- `resourceRef`

  The name of the resource.

`<resourcegroupResourceList>`
A list of the resources that are defined in a resource group.

Parent: `<resourcegroup>`

Children: Optional:

- `<resourcegroupResource>`

Attributes: None

`<resourceNode>`

The node on which a resource is defined.

Parent: `<resourceNodeList>`

Children: Required:

- `<state>`
- `<monitoredState>`

Optional:

- `<propertyList>`

Attributes: Required:

- `nodeRef`

  The name of the resource type.

Optional:

- `zone`

  The name of the zone.

`<resourcetype>`

A cluster resource type that is available in the cluster.

Parent: `<resourcetypeList>`

Children: Optional:

- `<resourcetypeRTRFile>`
- `<resourcetypeNodeList>`
- `<methodList>`
- `<parameterList>`
- `<propertyList>`

Attributes: Required:

- `name`

  The name of the resource type.

`<resourcetypeList>`

The root node of the cluster resource types that are defined in the configuration.

Parent: `<cluster>`

Children: Optional:

- `<resourcetype>`

Attributes: None

`<resourcetypeNode>`
A node on which a resource type is defined.

Parent: `<resourcetypeNodeList>`

Children: None

Attributes: Required:

- `nodeRef`

  The name of the cluster node.

`<resourcetypeNodeList>`
A list of the cluster nodes on which a particular resource type exists.

Parent: `<resourcetype>`

Children: Required : The `<resourcetypeNodeList>` element must contain either one or more `<resourcetypeNode>` elements or exactly one `<allNodes>` element.

- `<resourcetypeNode>`
- `<allNodes>`

Attributes: None

`<resourcetypeRTRFile>`
The name of a resource type registration (RTR) file that describes a particular resource type.

Parent: `<resourcetype>`

Children: None

Attributes: Required:

- `name`

  The name of the RTR file.

`<snmphost>`
The SNMP host and community that are configured on a cluster node.

Parent: `<snmphostList>`

Children: None

Attributes: Required:

- `community`

  The SNMP community name.

- `name`

  The name of the instance.

- `nodeRef`

  The node on which the SNMP host and community exist.

`<snmphostList>`
A list of the SNMP hosts and communities that are configured on a cluster node.

Parent: `<cluster>`

Children: Optional:

- `<snmphost>`

Attributes: None

`<snmpmib>`
An SNMP MIB that is on a cluster node.

Parent: `<snmpmibList>`

Children: Optional:

- `state`

Attributes: Required:

- `name`

  The name of the MIB.

- `nodeRef`

  The node on which the SNMP MIB exists.

Optional:

- `protocol`

  The SNMP protocol that the MIB will use. This attribute defaults to `SNMPv2`.

- `value`

  SNMPv3 or SNMPv2

`snmpmibList`
A list of the SNMP MIBs that are on a cluster node.

Parent: `<cluster>`

Children: Optional:

- `<snmpmib>`

Attributes: None

`<snmpuser>`
The SNMPv3 user that is configured on a cluster node.

Parent: `<snmpuserList>`

Children: None

Attributes: Required:

- `name`

  The name of the user.
- `nodeRef`

  The node on which the SNMPv3 user exists.
- `auth`

  The `auth` attribute can be set to `MD5` or `SHA`.

Optional:

- `defaultUser`

  The `defaultUser` attribute can be set to `yes` or `no`. If a value is not specified, the attribute defaults to whichever value is appropriate, based on the node configuration.
- `defaultSecurityLevel`

  The security level of the user. The `security` attribute can be set to one of the following values:

  - `authPriv`
  - `authNoPriv`
  - `noAuthNoPriv`

`<snmpuserList>`
A list of the SNMPv3 users that are configured on a cluster node.

Parent: `<cluster>`

Children: <snmpuser>

Attributes: None

<state>
The state of various objects within the cluster configuration. The <state> element is a generic element.

Parent: <quorumDevicePath>, <resourceNode>, <snmpmib>, <telemetryAttribute>, <transportAdapter>, <transportCable>, <transportSwitch>

Children: None

Attributes: Required:

- value

  The value attribute can be set to enabled or disabled.

<telemetrics>
Cluster monitoring thresholds

Parent: <cluster>

Children: Optional:

- <telemetryObjectType>

Attributes: None

<telemetryAttribute>
The attributes of system resources that you can monitor.

Parent: <telemetryObjectType>

Children: Required:

- <state> (1 or more)

Attributes: Required:

- name

  The name of the attribute.

<telemetryObjectType>
The types of objects you can monitor.

Parent: <telemetrics>

Children: Required:

- <telemetryAttibute>

Attributes: Required:

- name

  The name of the attribute.

`<transportAdapter>`

A network adapter that is used in the private cluster transport.

Parent: `<transportNode>`

Children: Optional:

- `<state>`
- `<transportType>`
- `<propertyList>`

Attributes: Required:

- `name`

  The name of the network adapter.

`<transportCable>`

A network cable that is used in the private cluster transport. The cable does not necessarily imply a physical cable, but rather a path between two `<endpoint>` elements.

Parent: `<transportCableList>`

Children: Required:

- `<endpoint>`

  The `<transportCable>` element must have two `<endpoint>` elements. Each `endpoint` element must describe one of the cable endpoints.

  Optional:

- `<state>`

  The `<transportCable>` element can have one `<state>` element.

Attributes: None

`<transportCableList>`

A list of the network cables that are used to connect two cluster `<endpoint>` elements.

Parent: `<clusterTransport>`

Children: `Optional:`

- `<transportCable>`

Attributes: None

`<transportNode>`
One of the cluster nodes that is used in the private cluster transport. Specify one `<transportNode>` element for each node of the cluster.

Parent: `<transportNodeList>`

Children: Optional:

- `<nodeRef>`

Attributes: Required:

- `transportAdapterList`

  The name of the cluster node.

`<transportNodeList>`
A list of the nodes that are used in the private cluster transport. This list of nodes always contains the same set of nodes as the members of the cluster.

Parent: `<clusterTransport>`

Children: Optional:

- `<transportNode>`

Attributes: None

`<transportSwitch>`
A cluster transport switch.

Parent: `<transportSwitchList>`

Children: Optional:

- `<state>`

Attributes: Required:

- `name`

  The name of the transport switch.

  Optional:

- `port`

  The number of the port on the switch.

`<transportSwitchList>`
A list of the network switches that are used by the private cluster transport system.

Parent: `<clusterTransport>`

Children:	Optional: ■ `<transportSwitch>`
Attributes:	None

`<transportType>`
The type of network transport that is used for a `<transportAdapter>` element.

Parent:	`<transportAdapter>`
Children:	Optional: ■ `<propertyList>`
Attributes:	Required: ■ `value` The `value` attribute can be set to `dlpi` or `rsm`.

Files `/usr/cluster/lib/xml/cluster.dtd`
The document type definition (DTD) file that defines the structure of the Sun Cluster configuration XML file.

See Also Intro(1CL), cluster(1CL)

http://www.w3.org/XML/

REFERENCE

SC327

Name clprivnet – SUNW,clprivnet Sun Cluster private network driver

Synopsis

```
/dev/clprivnet
```

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

The `SUNW,clprivnet` Sun Cluster private network driver is a STREAMS pseudo driver supporting Sun Cluster resident applications that use standard Solaris interfaces to communicate over the Sun Cluster private network. By striping data traffic over all links, this driver optimally utilizes the bandwidth of the private network while supporting highly available, software fault-tolerant communication.

Application Programming Interface The driver is supported by the character-special device `/dev/clprivnet`, but is reserved for Sun Cluster internal operation and the standard Solaris network utilities. This interface must not be directly used for general application communication.

Administration The administration and configuration of the driver as a network interface is done completely by the Sun Cluster infrastructure internals.

Files

`/dev/clprivnet`
: `clprivnet` special character device

`/usr/kernel/drv/clprivnet.conf`
: System-wide default device driver properties

Name did – user configurable disk id driver

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

Disk ID (DID) is a user configurable pseudo device driver that provides access to underlying disk, tape, and CDROM devices. When the device supports unique device ids, multiple paths to a device are determined according to the device id of the device. Even if multiple paths are available with the same device id, only one DID name is given to the actual device.

In a clustered environment, a particular physical device will have the same DID name regardless of its connectivity to more than one host or controller. This, however, is only true of devices that support a global unique device identifier such as physical disks.

DID maintains parallel directories for each type of device that it manages under `/dev/did`. The devices in these directories behave the same as their non-DID counterparts. This includes maintaining slices for disk and CDROM devices as well as names for different tape device behaviors. Both raw and block device access is also supported for disks by means of `/dev/did/rdsk` and `/dev/did/rdsk`.

At any point in time, I/O is only supported down one path to the device. No multipathing support is currently available through DID.

Before a DID device can be used, it must first be initialized by means of the scdidadm(1M) command.

ioctls The DID driver maintains an admin node as well as nodes for each DID device minor.

No user ioctls are supported by the admin node.

The `DKIOCINFO` ioctl is supported when called against the DID device nodes such as `/dev/did/rdsk/d0s2`.

All other ioctls are passed directly to the driver below.

Files

`/dev/did/dsk/dnsm`	block disk or CDROM device, where `n` is the device number and `m` is the slice number
`/dev/did/rdsk/dnsm`	raw disk or CDROM device, where `n` is the device number and `m` is the slice number
`/dev/did/rmt/n`	tape device , where `n` is the device number
`/dev/did/admin`	administrative device
`/kernel/drv/did`	driver module

`/kernel/drv/did.conf`	driver configuration file
`/etc/did.conf`	`scdidadm` configuration file for non-clustered systems
`Cluster Configuration Repository` (CCR) files	scdidadm(1M) maintains configuration in the CCR for clustered systems

See Also `devfsadm`(1M), Intro(1CL), cldevice(1CL), scdidadm(1M)

Notes DID creates names for devices in groups, in order to decrease the overhead during device hot-plug. For disks, device names are created in `/dev/did/dsk` and `/dev/did/rdsk` in groups of 100 disks at a time. For tapes, device names are created in `/dev/did/rmt` in groups of 10 tapes at a time. If more devices are added to the cluster than are handled by the current names, another group will be created.

REFERENCE

SC32 7p

Name sctransp_dlpi – configure the dlpi cluster interconnect

Description **Note –** Beginning with the Sun Cluster 3.2 release, Sun Cluster software includes an object-oriented command set. Although Sun Cluster software still supports the original command set, Sun Cluster procedural documentation uses only the object-oriented command set. For more information about the object-oriented command set, see the Intro(1CL) man page.

`dlpi` is a supported cluster transport type.

See Also Intro(1CL), cluster(1CL), scconf(1M), scinstall(1M)

Index

A

abort step timeout
  Oracle distributed lock manager (DLM), 754
  Solaris Volume Manager for Sun Cluster, 750
  VERITAS Volume Manager (VxVM), 744
access cluster information, 309
access resource group information, 332
access resource information, 326
add, change or update raw-disk device group configuration, 386
add/change/update VxVM device group configuration., 391
administer Sun Cluster SNMP hosts, 231
administer Sun Cluster SNMP MIB, 238
administer Sun Cluster SNMP users, 246
administer the private IP address range, 465
allocate and initialize DSDL environment, 566

C

callback interface for management of services as Sun Cluster resources, 296
cconsole, 342
ccp, 344
change Solaris Volume Manager device group configuration., 389
check for and report on vulnerable Sun Cluster configurations, 361
chosts, 346
cl_eventd, 347
claccess, 30
clconfiguration, 770
cldev, 35
cldevice, 35
cldevicegroup, 50
cldg, 50
clinterconnect, 71
clintr, 71
clmib, 238
clnas, 81
clnasdevice, 81
clnode, 93
clprivnet, 796
clq, 110
clqs, 123
clquorum, 110
clquorumserver, 123
clreslogicalhostname, 129
`clresource` command, 148
clresourcegroup, 176
`clresourcetype` command, 198
clressharedaddress, 210
clrg, 176
`clrs` command, 148
clrslh, 129
clrssa, 210
`clrt` command, 198
clsetup, 229
clsnmphost, 231
clsnmpmib, 238
clsnmpuser, 246
clta, 254
cltelemetryattribute, 254

cluster, 269
cluster event daemon, 347
cluster feature, VERITAS Volume Manager (VxVM), 743
cluster information access functions., 613
cluster log facility access, 608
cluster names database, 646
Cluster Ready Services (CRS), framework, 711
clusters, 646
clvxvm, 291
command log file`commandlog`, 648
command standard output for scha_cluster_get, scha_control, scha_resource_get, scha_resourcegroup_get, scha_resourcetype_get, scha_resource_setstatus, 312
`commandlog`, 648
communications ports
  UNIX Distributed Lock Manager (Oracle UDLM), 753
  VERITAS Volume Manager (VxVM), 745
configuration daemon, VERITAS Volume Manager (VxVM), 745
configuration files, Oracle distributed lock manager (DLM), 753
configure an Ethernet cluster transport switch, 411
configure an InfiniBand cluster transport switch, 412
configure resource type template, 303
configure Sun Cluster interactively, 229
configure system resource monitoring, 254
configure the dlpi cluster interconnect, 800
configure the Dolphin cluster transport switch, 410
configure the eri transport adapter, 404
configure the Gigabit Ethernet (ge) transport adapter, 405
configure the hme transport adapter, 406
configure the InfiniBand (ibd) transport adapter, 407
configure the Intel PRO/1000 network adapter, 403
configure the qfe transport adapter, 408
configure the SCI–PCI cluster transport adapter, 409
configure VERITAS Volume Manager for Sun Cluster, 291
cports, 348
create a Sun Cluster resource type template, 306
crlogin, 342
CRS (Cluster Ready Services), framework, 711
`crs_framework` resource type, 711
cssh, 342
ctelnet, 342
`cvm_abort_step_timeout` extension property, 744
`cvm_return_step_timeout` extension property, 744
`cvm_start_step_timeout` extension property, 744
`cvm_step1_timeout` extension property, 744
`cvm_step2_timeout` extension property, 744
`cvm_step3_timeout` extension property, 744
`cvm_step4_timeout` extension property, 744
`cvm_stop_step_timeout` extension property, 745

D

daemons
  `vxconfigd`, 745
  `vxkmsgd`, 745
`debug_level` extension property
  `rac_svm` resource type, 749
`debug_Level` extension property
  `ScalDeviceGroup` resource type, 757
  `ScalMountPoint` resource type, 762
`debug_level` extension property
  `SUNW.rac_svm` resource type, 749
`debug_Level` extension property
  `SUNW.ScalDeviceGroup` resource type, 757
  `SUNW.ScalMountPoint` resource type, 762
derby, resource type implementation of Java DB database, 714
Derby database, resource type implementation, 728
determine if a PMF-monitored process tree exists, 568
device groups
  scalable
    resource type for, 756
device identifier configuration and administration utility wrapper, 413
did, 797
disk path monitoring administration command, 422
`diskgroupname` extension property, 757
distributed lock manager (DLM), *See* Oracle distributed lock manager (DLM)
DLM (distributed lock manager), *See* Oracle distributed lock manager (DLM)

DSDL (Data Service Development Library)
fault monitor functions, 526
fault monitoring, 524
network resource access functions, 524
Process Monitor Facility (PMF) functions, 525
property functions, 524
utility functions, 526

E

establish a TCP connection to an application, 533
establish a tcp connection to an application, 541
event, 716
execute a given command in a given amount of time, 600
execute a program under PMF control, 573
expand cluster names into host names, 346
extension properties
`crs_framework` resource type, 713
`rac_cvm` resource type, 744
`rac_framework` resource type, 747
`rac_svm` resource type, 749
`rac_udlm` resource type, 752
`SUNW.crs_framework` resource type, 713
`SUNW.rac_cvm` resource type, 744
`SUNW.rac_framework` resource type, 747
`SUNW.rac_svm` resource type, 749
`SUNW.rac_udlm` resource type, 752

F

`failfastmode` extension property, 753
failover a resource group, 529
fault monitor, functions, DSDL, 526
file systems
scalable
resource type for, 761
`filesystemtype` extension property, 762
frameworks
CRS, 711
Sun Cluster Support for Oracle RAC, 747
free DSDL environment resources, 527
free the network address memory, 549
free the network resource memory, 550
free the port list memory, 551
free the resource extension property memory, 548
functions
DSDL fault monitor, 526
DSDL network resource access, 524
DSDL Process Monitor Facility (PMF), 525
DSDL property, 524
DSDL utility, 526

G

generate an error string from an error code, 528
generate error message from error code, 642
generate error string from error code, 642
get status information about SUNW.HAStoragePlus resources that are used by a resource, 564
get the network addresses used by a resource, 554
get the network resources used by a resource, 561
get the network resources used in a resource group, 560
global devices namespace administration script, 432

H

HADerby, resource type implementation of Derby database, 728
halockrun, 349
hatimerun, 351

I

initialize Sun Cluster software and establish new cluster nodes, 434
initialize system resource monitoring, 508
Install VERITAS Volume Manager (VxVM) on a cluster node., 514
interactive cluster configuration tool, 481
intro, 16
Intro, 16
introduction to Sun Cluster maintenance commands, 16

`iotimeout` extension property, 763

J

Java DB database, resource type implementation, 714

L

launch GUI version of Sun Cluster Agent Builder, 302
libschost.so.1, 24
local cluster node name access function, 610
`logicaldevicelist` extension property, 758

M

manage access to NAS devices for Sun Cluster, 81
manage quorum servers, 123
manage registration and unregistration of resource types, resource groups, and resources., 471
manage resource groups for Sun Cluster data services, 176
manage resources for Sun Cluster logical hostnames, 129
manage Sun Cluster access policies for adding nodes, 30
manage Sun Cluster device groups, 50
manage Sun Cluster devices, 35
manage Sun Cluster event Management Information Base (MIB) module, 426
manage Sun Cluster nodes, 93
manage Sun Cluster quorum, 110
manage Sun Cluster resources for shared addresses, 210
manage the global configuration and status of a cluster, 269
manage the Sun Cluster interconnect, 71
managing
  resource types, 198
  resources, 148
messaging daemon, VERITAS Volume Manager (VxVM), 745
`monitor_retry_count` extension property
  `ScalDeviceGroup` resource type, 759
  `ScalMountPoint` resource type, 763
  `SUNW.ScalDeviceGroup` resource type, 759
  `SUNW.ScalMountPoint` resource type, 763
`monitor_retry_interval` extension property
  `ScalDeviceGroup` resource type, 759
  `ScalMountPoint` resource type, 763
  `SUNW.ScalDeviceGroup` resource type, 759
  `SUNW.ScalMountPoint` resource type, 763
monitoring
  Sun Cluster Support for Oracle RAC, 747
  UNIX Distributed Lock Manager (Oracle UDLM, 752
monitoring the status of a Sun Cluster configuration, 488
mount points
  scalable
    resource type for, 761
`mountoptions` extension property, 763
`mountpointdir` extension property, 764
multi window, multi machine, remote console, login, ssh, and telnet commands, 342

N

NAS (network-attached storage devices, 761
network-attached storage (NAS) devices, 761
network resource access functions, DSDL, 524
`num_ports` extension property, 753

O

Oracle Cluster Ready Services (CRS), framework, 711
`oracle_config_file` extension property, 753
Oracle CRS (Cluster Ready Services), framework, 711
Oracle distributed lock manager (DLM), 753
Oracle Parallel Server, *See* Sun Cluster Support for Oracle RAC
Oracle RAC (Real Application Clusters), *See* Sun Cluster Support for Oracle RAC
Oracle Real Application Clusters (RAC), *See* Sun Cluster Support for Oracle RAC

Oracle UDLM (UNIX Distributed Lock Manager), 752

P

perform ownership/state change of resource groups and device groups in Sun Cluster configurations, 493
PMF (Process Monitor Facility), functions, DSDL, 525
pmfadm, 353
pmfd, 360
pnmd, 358
`port` extension property, 753
ports, *See* communications ports
print the contents of a list of hostname-port-protocol 3-tuples used by a resource group, 579
print the contents of a network resource list, 580
print the contents of a port list, 581
probe by establishing and terminating a TCP connection to an application, 590, 592
process monitor facility administration, 353
programs, `vxclust`, 745
Public Network Management (PNM) service daemon, 358

Q

QFS shared file system, *See* Sun StorEdge QFS shared file system

R

r_properties, 679
RAC (Real Application Clusters), *See* Sun Cluster Support for Oracle RAC
`rac_cvm` resource type, 743
`rac_framework` resource type, 747
`rac_svm` resource type, 749
`rac_udlm` resource type, 752
rdt_setmtu, 359
read data using a tcp connection to an application, 544
Real Application Clusters (RAC), *See* Sun Cluster Support for Oracle RAC
reconfiguration timeouts
  Oracle distributed lock manager (DLM), 754
  reservation step, 747
  Solaris Volume Manager for Sun Cluster, 750
  VERITAS Volume Manager (VxVM), 744
request resource and resource group control, 320
reservation step timeout, 747
`reservation_timeout` extension property, 747
resource and resource group control request function, 618
resource group properties, 666
Resource information access command, 326
resource information access functions, 624, 628
resource properties, 679
resource type, 721
resource type for data collection, system resource usage, 767
resource type for proxying failover SMF services, 734
resource type implementation for the Cluster Reconfiguration Notification Protocol (CRNP), 716
resource type information access command, 338
resource type information access functions, 638
resource-type properties, 699
resource type registration (RTR) file, 650
resource type that enforces dependencies between Sun Cluster device services, file systems, and data services, 730
resource types
  `crs_framework`, 711
  managing, 198
  `rac_cvm`, 743
  `rac_framework`, 747
  `rac_svm`, 749
  `rac_udlm`, 752
  `ScalDeviceGroup`, 756
  `ScalMountPoint`, 761
  `SUNW.crs_framework`, 711
  `SUNW.rac_cvm`, 743
  `SUNW.rac_framework`, 747
  `SUNW.rac_svm`, 749
  `SUNW.rac_udlm`, 752
  `SUNW.ScalDeviceGroup`, 756
  `SUNW.ScalMountPoint`, 761
resources, managing, 148

restart a resource, 588
restart a resource group, 589
restart fault monitor using PMF, 570
restrictions
  rac_cvm resource type, 744
  rac_udlm resource type, 752
  SUNW.rac_cvm resource type, 744
  SUNW.rac_udlm resource type, 752
retrieve an extension property, 552
retrieve configuration data about resource groups, resource types, and resources, 485
retrieve the name of a zone on whose behalf a method is running, 562
retrieve the port list used by a resource, 556
retrieve the resource group name, 557
retrieve the resource name, 558
retrieve the resource type name, 559
return step timeout
  Solaris Volume Manager for Sun Cluster, 750
  VERITAS Volume Manager (VxVM), 744
rg_properties, 666
RPC-based process monitor server, 360
rpc.pmfd, 360
rt_callbacks, 296
rt_properties, 699
rt_reg, 650
run a child program while holding a file lock, 349
run child program under a timeout, 351

S

sc_zonesd, 519
sc_zonesd - Sun Cluster zone administration daemonSun Cluster Private IP address service daemon, 519
scalable device groups, resource type for, 756
scalable file systems, resource type for, 761
scalable mount points, resource type for, 761
ScalDeviceGroup resource type, 756
ScalMountPoint resource type, 761
sccheck, 361
sccheckd, 365
scconf, 366
scconf_dg_rawdisk, 386
scconf_dg_svm, 389
scconf_dg_vxvm, 391
scconf_transp_adap_e1000g, 403
scconf_transp_adap_eri, 404
scconf_transp_adap_ge, 405
scconf_transp_adap_hme, 406
scconf_transp_adap_ibd, 407
scconf_transp_adap_qfe, 408
scconf_transp_adap_sci, 409
scconf_transp_jct_dolphinswitch, 410
scconf_transp_jct_etherswitch, 411
scconf_transp_jct_ibswitch, 412
scdidadm, 413
scdpm, 422
scds_calls, 522
scds_close, 527
scds_error_string, 528
scds_error_string_i18n, 528
scds_failover_rg, 529
scds_fm_action, 530
scds_fm_net_connect, 533
scds_fm_net_disconnect, 536
scds_fm_sleep, 539
scds_fm_tcp_connect, 541
scds_fm_tcp_disconnect, 543
scds_fm_tcp_read, 544
scds_fm_tcp_write, 546
scds_free_ext_property, 548
scds_free_net_list, 550
scds_free_netaddr_list, 549
scds_free_port_list, 551
scds_get_ext_property, 552
scds_get_netaddr_list, 554
scds_get_port_list, 556
scds_get_resource_group_name, 557
scds_get_resource_name, 558
scds_get_resource_type_name, 559
scds_get_rg_hostnames, 560
scds_get_rs_hostnames, 561
scds_get_zone_name, 562
scds_hasp_check, 564
scds_initialize, 566
scds_pmf_get_status, 568
scds_pmf_restart_fm, 570

scds_pmf_signal, 571
scds_pmf_start, 573
scds_pmf_stop, 575
scds_pmf_stop_monitoring, 577
scds_print_net_list, 580
scds_print_netaddr_list, 579
scds_print_port_list, 581
scds_restart_resource, 588
scds_restart_rg, 589
scds_simple_net_probe, 590
scds_simple_probe, 592
scds_svc_wait, 594
scds_syslog, 597
scds_syslog_debug, 598
scds_timerun, 600
scdsbuilder, 302
scdsconfig, 303
scdscreate, 306
sceventmib, 426
scgdevs, 432
scha_calls, 602
scha_cluster_close, 613
scha_cluster_get, 309, 613
scha_cluster_getlogfacility, 608
scha_cluster_getnodename, 610
scha_cluster_getzone, 611
scha_cluster_open, 613
scha_cmds, 312
scha_control, 320, 602, 618
scha_control_zone, 618
scha_get_function, 602
scha_resource_close, 628
scha_resource_get, 326, 628
scha_resource_open, 628
scha_resource_setstatus, 335, 636
scha_resource_setstatus_zone, 636
scha_resourcegroup_close, 624
scha_resourcegroup_get, 332, 624
scha_resourcegroup_open, 624
scha_resourcetype_close, 638
scha_resourcetype_get, 338, 638
scha_resourcetype_open, 638
scha_strerror, 642
scha_strerror_i18n, 642
`schedclass` extension property, 753
`schedpriority` extension property, 753
scinstall, 434
scprivipadm, 465
scprivipd - Sun Cluster Private IP address service daemon, 469
scqsd.conf, 658
scrgadm, 471
scsetup, 481
scshutdown, 483
scsnapshot, 485
scstat, 488
scswitch, 493
sctelemetry, 508
SCTelemetry, resource type for collecting data on system resource usage, 767
sctransp_dlpi, 800
scversions, 512
scvxinstall, 514
send a signal to a process tree under PMF control, 571
service for the cluster configuration check utility, 365
set resource status, 335
set resource status function, 636
set the MTU size in RSMRDT driver, 359
shared object to provide logical host name instead of physical host name, 24
shut down a cluster, 483
Solaris Volume Manager, 749, 756
start step timeout
  Oracle distributed lock manager (DLM), 754
  Solaris Volume Manager for Sun Cluster, 750
  VERITAS Volume Manager (VxVM), 744
startup file for the quorum server, 658
status information
  Sun Cluster Support for Oracle RAC, 747
  UNIX Distributed Lock Manager (Oracle UDLM, 752
stop monitoring a process that is running under PMF control, 577
Sun Cluster Data Services Development Library (DSDL) functions, 522
Sun Cluster library functions used in the implementation of callback methods and monitors of resource types, 602

Sun Cluster Private IP address service daemon, 469
Sun Cluster Support for Oracle RAC
  CRS framework, 711
  device groups, 759
  file systems, 766
  framework, 747
  monitoring, 747
  resource types
    crs_framework, 711
    rac_cvm, 743
    rac_framework, 747
    rac_svm, 749
    rac_udlm, 752
    SUNW.crs_framework, 711
    SUNW.rac_cvm, 743
    SUNW.rac_framework, 747
    SUNW.rac_svm, 749
    SUNW.rac_udlm, 752
  Solaris Volume Manager, 749
  status information, 747
  UNIX Distributed Lock Manager (Oracle UDLM), 752
  VERITAS Volume Manager (VxVM), 743
Sun Cluster system configuration file, 770
Sun Cluster version management, 512
Sun StorEdge QFS shared file system, 761
SUNW,clprivnet Sun Cluster private network driver, 796
SUNW.crs_framework resource type, 711
SUNW.derby, resource type implementation of Java DB database, 714
SUNW.Event, 716
SUNW.gds, 721
SUNW.HADerby, resource type implementation of Derby database, 728
SUNW.HAStoragePlus, 730
SUNW.proxysmffailoverSUNW.Proxy_SMF_failover, 734
SUNW.rac_cvm resource type, 743
SUNW.rac_framework resource type, 747
SUNW.rac_svm resource type, 749
SUNW.rac_udlm resource type, 752
SUNW.ScalDeviceGroup resource type, 756
SUNW.ScalMountPoint resource type, 761
SUNW.SCTelemetry, resource type for collecting data on system resource usage, 767
svm_abort_step_timeout extension property, 750
svm_return_step_timeout extension property, 750
svm_start_step_timeout extension property, 750
svm_step1_timeout extension property, 750
svm_step2_timeout extension property, 750
svm_step3_timeout extension property, 750
svm_step4_timeout extension property, 750
svm_stop_step_timeout extension property, 750
system resource usage, resource type for data collection, 767

T

take action after probe completion function, 530
targetfilesystem extension property, 764
TCP connections, using DSDL fault monitoring, 524
terminate a process that is running under PMF control, 575
terminate a TCP connection to an application, 536
terminate a tcp connection to an application, 543
the Sun Cluster System Cluster Control Panel GUI, 344
timeouts
  Oracle distributed lock manager (DLM), 754
  reservation step, 747
  Solaris Volume Manager for Sun Cluster, 750
  VERITAS Volume Manager (VxVM), 744
triples">expand host names into <host, server, port> triples, 348

U

udlm_abort_step_timeout extension property, 754
udlm.conf configuration file, 753
udlm_start_step_timeout extension property, 754
udlm_step1_timeout extension property, 754
udlm_step2_timeout extension property, 754
udlm_step3_timeout extension property, 754
udlm_step4_timeout extension property, 754
udlm_step5_timeout extension property, 754
UNIX Distributed Lock Manager (Oracle UDLM), 752

update the Sun Cluster software configuration, 366
upgrade configuration data about resource groups, resource types, and resources, 485
user configurable disk id driver, 797
utility functions, DSDL, 526

V

VERITAS Volume Manager (VxVM), 743, 756
volumes, 758
`vxclust_num_ports` extension property, 745
`vxclust_port` extension property, 745
`vxclust` program, 745
`vxconfigd` daemon, 745
`vxconfigd_port` extension property, 745
`vxkmsgd` daemon, 745
`vxkmsgd_port` extension property, 745
VxVM (VERITAS Volume Manager), 743, 756

W

wait for a message on a fault monitor control socket, 539
wait for the specified timeout period for a monitored process to die, 594
write a debugging message to the system log, 598
write a message to the system log, 597
write data using a tcp connection to an application, 546

Z

zone name access function, 611