Sun Cluster Data Services Planning and Administration Guide for Solaris OS

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

Part No: 820–7359–10
November 2009, Revision A

091123@23031

Contents

Preface

Sun Cluster Data Services Planning and Administration Guide for Solaris OS explains how to install and configure Sun™ Cluster data services.

Note – This Sun Cluster release supports systems that use the SPARC and x86 families of processor architectures: UltraSPARC, SPARC64, AMD64, and Intel 64. In this document, x86 refers to the larger family of 64-bit x86 compatible products. Information in this document pertains to all platforms unless otherwise specified.

This document is intended for system administrators with extensive knowledge of Sun software and hardware. Do not use this document as a planning or presales guide. Before reading this document, you should have already determined your system requirements and purchased the appropriate equipment and software.

The instructions in this book assume knowledge of the Solaris™ Operating System (Solaris OS) and expertise with the volume-manager software that is used with Sun Cluster software.

Using UNIX Commands

This document contains information about commands that are specific to installing and configuring Sun Cluster data services. The document does *not* contain comprehensive information about basic UNIX® commands and procedures, such as shutting down the system, booting the system, and configuring devices. Information about basic UNIX commands and procedures is available from the following sources:

- Online documentation for the Solaris Operating System
- Solaris Operating System man pages
- Other software documentation that you received with your system

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

Related Documentation

Information about related Sun Cluster topics is available in the documentation that is listed in the following table. All Sun Cluster documentation is available at `http://docs.sun.com`.

Topic	Documentation
Data service administration	*Sun Cluster Data Services Planning and Administration Guide for Solaris OS*
	Individual data service guides
Concepts	*Sun Cluster Concepts Guide for Solaris OS*
Overview	*Sun Cluster Overview for Solaris OS*
Software installation	*Sun Cluster Software Installation Guide for Solaris OS*
System administration	*Sun Cluster System Administration Guide for Solaris OS*
Hardware administration	*Sun Cluster 3.1 - 3.2 Hardware Administration Manual for Solaris OS*
	Individual hardware administration guides
Data service development	*Sun Cluster Data Services Developer's Guide for Solaris OS*
Error messages	*Sun Cluster Error Messages Guide for Solaris OS*
Command and function reference	*Sun Cluster Reference Manual for Solaris OS*

For a complete list of Sun Cluster documentation, see the release notes for your release of Sun Cluster at `http://docs.sun.com`.

Related Third-Party Web Site References

Third-party URLs that are referenced in this document provide additional related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to `http://docs.sun.com` and click Feedback.

Getting Help

If you have problems installing or using Sun Cluster, contact your service provider and provide the following information:

- Your name and email address (if available)
- Your company name, address, and phone number
- The model number and serial number of your systems
- The release number of the Solaris Operating System (for example, Solaris 10)
- The release number of Sun Cluster (for example, Sun Cluster 3.2)

Use the following commands to gather information about each node on your system for your service provider.

Command	Function
`prtconf -v`	Displays the size of the system memory and reports information about peripheral devices
`psrinfo -v`	Displays information about processors
`showrev –p`	Reports which patches are installed
`prtdiag -v`	Displays system diagnostic information
`/usr/cluster/bin/clnode show-rev`	Displays Sun Cluster release and package version information

Also have available the contents of the `/var/adm/messages` file.

◆ ◆ ◆ CHAPTER 1

Planning for Sun Cluster Data Services

This chapter provides planning information and guidelines to install and configure Sun Cluster data services. This chapter contains the following sections.

For information about data services, resource types, resources, and resource groups, see *Sun Cluster Concepts Guide for Solaris OS*.

Sun Cluster software can provide service only for those data services that are either supplied with the Sun Cluster product or are created with the Sun Cluster data services application programming interfaces (APIs).

If a Sun Cluster data service is not provided for your application, consider developing a custom data service for the application. To develop a custom data service, use the Sun Cluster data services APIs. For more information, see *Sun Cluster Data Services Developer's Guide for Solaris OS*.

Note – Sun Cluster does not provide a data service for the `sendmail`(1M) subsystem. The `sendmail` subsystem can run on the individual cluster nodes, but the `sendmail` functionality is not highly available. This restriction applies to all the `sendmail` functionality, including the functionality of mail delivery and mail routing, queuing, and retry.

Configuration Guidelines for Sun Cluster Data Services

This section provides configuration guidelines for Sun Cluster data services.

Identifying Data Service Special Requirements

Identify requirements for all of the data services **before** you begin Solaris and Sun Cluster installation. Failure to do so might result in installation errors that require that you completely reinstall the Solaris and Sun Cluster software.

For example, the Oracle Real Application Clusters Guard option of Sun Cluster Support for Oracle Real Application Clusters has special requirements for the hostnames that you use in the cluster. Sun Cluster HA for SAP also has special requirements. You must accommodate these requirements before you install Sun Cluster software because you cannot change hostnames after you install Sun Cluster software.

Note – Some Sun Cluster data services are not supported for use in x86 based clusters. For more information, see the release notes for your release of Sun Cluster at `http://docs.sun.com`.

Determining the Location of the Application Binaries

You can install the application software and application configuration files on one of the following locations.

- **The local disks of each cluster node –** Placing the software and configuration files on the individual cluster nodes provides the following advantage. You can upgrade application software later without shutting down the service.

  The disadvantage is that you then have several copies of the software and configuration files to maintain and administer.
- **The cluster file system –** If you put the application binaries on the cluster file system, you have only one copy to maintain and manage. However, you must shut down the data service in the entire cluster to upgrade the application software. If you can spare a short period of downtime for upgrades, place a single copy of the application and configuration files on the cluster file system.

  For information about how to create cluster file systems, see "Planning the Global Devices, Device Groups, and Cluster File Systems" in *Sun Cluster Software Installation Guide for Solaris OS*.
- **Highly available local file system –** Using `HAStoragePlus`, you can integrate your local file system into the Sun Cluster environment, making the local file system highly available. `HAStoragePlus` provides additional file system capabilities such as checks, mounts, and unmounts that enable Sun Cluster to fail over local file systems. To fail over, the local file system must reside on global disk groups with affinity switchovers enabled.

For information about how to use the `HAStoragePlus` resource type, see "Enabling Highly Available Local File Systems" on page 110.

Verifying the `nsswitch.conf` File Contents

The `nsswitch.conf` file is the configuration file for name-service lookups. This file determines the following information.

- The databases within the Solaris environment to use for name-service lookups
- The order in which the databases are to be consulted

Some data services require that you direct "group" lookups to "files" first. For these data services, change the "group" line in the `nsswitch.conf` file so that the "files" entry is listed first. See the documentation for the data service that you plan to configure to determine whether you need to change the "group" line.

For additional information about how to configure the `nsswitch.conf` file for the Sun Cluster environment, see "Planning the Sun Cluster Environment" in *Sun Cluster Software Installation Guide for Solaris OS*.

Planning the Cluster File System Configuration

Depending on the data service, you might need to configure the cluster file system to meet Sun Cluster requirements. To determine whether any special considerations apply, see the documentation for the data service that you plan to configure.

For information about how to create cluster file systems, see "Planning the Global Devices, Device Groups, and Cluster File Systems" in *Sun Cluster Software Installation Guide for Solaris OS*.

The resource type `HAStoragePlus` enables you to use a highly available local file system in a Sun Cluster environment that is configured for failover. For information about setting up the `HAStoragePlus` resource type, see "Enabling Highly Available Local File Systems" on page 110.

Enabling Solaris SMF Services to Run Under the Control of Sun Cluster

The Service Management Facility (SMF) enables you to automatically start and restart SMF services, during a node boot or service failure. This feature is similar to the Sun Cluster Resource Group Manager (RGM), which facilitates high availability and scalability for cluster applications. SMF services and RGM features are complementary to each other.

Sun Cluster includes three new SMF proxy resource types that can be used to enable SMF services to run with Sun Cluster in a failover, multi-master, or scalable configuration. The SMF proxy resource types enables you to encapsulate a set of interrelated SMF services into a single resource, *SMF proxy resource* to be managed by Sun Cluster. In this feature, SMF manages the availability of SMF services on a single node. Sun Cluster provides cluster-wide high availability and scalability of the SMF services.

For information about how to encapsulate these services, see "Enabling Solaris SMF Services to Run With Sun Cluster" on page 147

You might require Sun Cluster to make highly available an application other than NFS or DNS that is integrated with the Solaris Service Management Facility (SMF). To ensure that Sun Cluster can restart or fail over the application correctly after a failure, you must disable SMF service instances for the application as follows:

- For any application other than NFS or DNS, disable the SMF service instance on all potential primary nodes for the Sun Cluster resource that represents the application.
- If multiple instances of the application share any component that you require Sun Cluster to monitor, disable all service instances of the application. Examples of such components are daemons, file systems, and devices.

Note – If you do not disable the SMF service instances of the application, both the Solaris SMF and Sun Cluster might attempt to control the startup and shutdown of the application. As a result, the behavior of the application might become unpredictable.

For more information, see the following documentation:

- "How to Disable a Service Instance" in *System Administration Guide: Basic Administration*
- *Sun Cluster Data Service for NFS Guide for Solaris OS*
- *Sun Cluster Concepts Guide for Solaris OS*

Relationship Between Resource Groups and Device Groups

Sun Cluster uses the concept of **node lists** for device groups and resource groups. Node lists are ordered lists of primary nodes, which are potential masters of the disk device group or resource group. Sun Cluster uses a **failback policy** to determine the behavior of Sun Cluster in response to the following set of conditions:

- A node that has failed and left the cluster rejoins the cluster.
- The node that is rejoining the cluster appears earlier in the node list than the current primary node.

If failback is set to `True`, the device group or resource group is switched off the current primary and switched onto the rejoining node, making the rejoining node the new primary.

For example, assume that you have a disk device group, `disk-group-1`, that has nodes `phys-schost-1` and `phys-schost-2` in its node list, with the failback policy set to `Enabled`. Assume that you also have a failover resource group, `resource-group-1`, which uses `disk-group-1` to hold its application data. When you set up `resource-group-1`, also specify `phys-schost-1` and `phys-schost-2` for the resource group's node list, and set the failback policy to `True`.

To ensure high availability of a scalable resource group, make the scalable resource group's node list a superset of the node list for the disk device group. This setting ensures that the nodes that are directly connected to the disks are also nodes that can run the scalable resource group. The advantage is that, when at least one cluster node connected to the data is up, the scalable resource group runs on that same node, making the scalable services available also.

For more information about the relationship between device groups and resource groups, see "Device Groups" in *Sun Cluster Overview for Solaris OS*.

For information about how to set up device groups, see the following documentation:

- "Device Groups" in *Sun Cluster Software Installation Guide for Solaris OS*

Understanding `HAStoragePlus`

The `HAStoragePlus` resource type can be used to configure the following options.

- Coordinate the boot order of disk devices and resource groups. Other resources in the resource group that contains the `HAStoragePlus` resource are brought online *only* after the disk device resources become available.
- With `AffinityOn` set to `True`, enforce collocation of resource groups and device groups on the same node. This enforced collocation enhances the performance of disk-intensive data services.

In addition, `HAStoragePlus` is capable of mounting local and global file systems. For more information, see "Planning the Cluster File System Configuration" on page 15.

Note – If the device group is switched to another node while the `HAStoragePlus` resource is online, `AffinityOn` has no effect. The resource group does **not** migrate with the device group. However, if the resource group is switched to another node, the setting of `AffinityOn` to `True` causes the device group to follow the resource group to the new node.

See "Synchronizing the Startups Between Resource Groups and Device Groups" on page 104 for information about the relationship between device groups and resource groups.

See "Enabling Highly Available Local File Systems" on page 110 for procedures for mounting of file systems such as VxFS and Solaris ZFS (Zettabyte File System) in a local mode. The `SUNW.HAStoragePlus(5)` man page provides additional details.

Determining Whether Your Data Service Requires `HAStoragePlus`

The following types of data services require `HAStoragePlus`:

- Data services with nodes that are not directly connected to storage
- Data services that are disk intensive

Data Services With Nodes That Are Not Directly Connected to Storage

Some nodes in the node list of a data service's resource group might not be directly connected to the storage. In this situation, you must coordinate the boot order between the storage and the data service. To meet this requirement, configure the resource group as follows:

- Configure `HAStoragePlus` resources in the resource group.
- Set the dependency of the other data service resources to the `HAStoragePlus` resource.

Data Services That Are Disk Intensive

Some data services, such as Sun Cluster HA for Oracle and Sun Cluster HA for NFS are disk intensive. If your data service is disk intensive, ensure that the resource groups and device groups are collocated on the same node. To meet this requirement, perform the following tasks.

- Adding an `HAStoragePlus` resource to your data service resource group
- Switching the `HAStoragePlus` resource online
- Setting the dependency of your data service resources to the `HAStoragePlus` resource
- Setting `AffinityOn` to `True`

Note – The failback settings must be identical for both the resource group and device groups.

Some data services are not disk intensive. For example, Sun Cluster HA for DNS, which reads all of its files at startup, is not disk intensive. If your data service is **not** disk intensive, configuring the `HAStoragePlus` resource type is optional.

Considerations for Installing and Configuring a Data Service

Use the information in this section to plan the installation and configuration of any data service. The information in this section encourages you to think about the impact your decisions have on the installation and configuration of any data service. For specific considerations for a data service, see the documentation for the data service.

- Retries within the I/O subsystem during disk failures might cause applications whose data services are disk intensive to experience delays. Disk-intensive data services are I/O intensive and have a large number of disks configured in the cluster. An I/O subsystem might require several minutes to retry and recover from a disk failure. This delay can cause Sun Cluster to fail over the application to another node, even though the disk might have eventually recovered on its own. To avoid failover during these instances, consider increasing the default probe timeout of the data service. If you need more information or help with increasing data service timeouts, contact your local support engineer.
- For better performance, install and configure your data service on the cluster nodes with direct connection to the storage.
- Client applications that run on cluster nodes should not map to logical IP addresses of an HA data service. After a failover, these logical IP addresses might no longer exist, leaving the client without a connection.

Node List Properties

You can specify the following node list properties when configuring data services.

- `installed_nodes` property
- `nodelist` property
- `auxnodelist` property

`installed_nodes` Property

The `installed_nodes` property is a property of the resource type for the data service. This property is a list of the cluster node names on which the resource type is installed and enabled to run.

`nodelist` Property

The `nodelist` property is a property of a resource group. This property specifies a list of cluster node names where the group can be brought online, in order of preference. These nodes are known as the potential primaries or masters of the resource group. For failover services, configure only one resource group node list. For scalable services, configure two resource groups and thus two node lists. One resource group and its node list identify the nodes on which the shared addresses are hosted. This list is a failover resource group on which the scalable resources depend. The other resource group and its list identify nodes on which the application resources are hosted. The application resources depend on the shared addresses. Therefore, the node list for the resource group that contains the shared addresses must be a superset of the node list for the application resources.

`auxnodelist` Property

The `auxnodelist` property is a property of a shared address resource. This property is a list of node IDs that identify cluster nodes that can host the shared address but never serve as primary in the case of failover. These nodes are mutually exclusive with the nodes that are identified in the node list of the resource group. This list pertains to scalable services only. For details, see the `clressharedaddress`(1CL) man page.

Overview of the Installation and Configuration Process

Use the following procedures to install and configure a data service.

- Install the data service packages from the installation medium on which the packages are supplied.
  - Sun Cluster 3.0 5/02 CD-ROM
  - Sun Cluster 3.0 5/02 Agents CD-ROM
- Install and configure the application to run in the cluster environment.
- Configure the resources and resource groups that the data service uses. When you configure a data service, specify the resource types, resources, and resource groups that the Resource Group Manager (RGM) is to manage. The documentation for the individual data services describes these procedures.

Before you install and configure data services, see *Sun Cluster Software Installation Guide for Solaris OS*, which includes instructions for the following tasks:

- Installing the data service software packages
- Configuring IP network multipathing groups that the network resources use

---

Note – You can use Sun Cluster Manager to install and configure the following data services: Sun Cluster HA for Oracle, Sun Cluster HA for Sun Java™ System Web Server, Sun Cluster HA for Web Server, Sun Cluster HA for Apache, Sun Cluster HA for DNS, and Sun Cluster HA for NFS. See the SunPlex Manager online help for more information.

Installation and Configuration Task Flow

The following table summarizes the tasks for installing and configuring Sun Cluster data services. The table also provides cross-references to detailed instructions for performing the tasks.

TABLE 1–1 Tasks for Installing and Configuring Sun Cluster Data Services

Task	Instructions
Install the Solaris and Sun Cluster software	*Sun Cluster Software Installation Guide for Solaris OS*
Set up IPMP groups	*Sun Cluster Software Installation Guide for Solaris OS*
Set up multihost disks	*Sun Cluster Software Installation Guide for Solaris OS*
Plan resources and resource groups	Appendix D, “Data Service Configuration Worksheets and Examples”
Decide the location for application binaries, and configure the `nsswitch.conf` file	“Determining the Location of the Application Binaries” on page 14 “Verifying the `nsswitch.conf` File Contents” on page 15
Install and configure the application software	The appropriate Sun Cluster data services book
Install the data service software packages	*Sun Cluster Software Installation Guide for Solaris OS* or the appropriate Sun Cluster data services book
Register and configure the data service	The appropriate Sun Cluster data services book

Example of Configuring a Failover Data Service

This example summarizes how to set up the resource types, resources, and resource groups that a failover data service for the Oracle application requires. For complete instructions for configuring the data service for the Oracle application, see *Sun Cluster Data Service for Oracle Guide for Solaris OS*.

The principal difference between this example and an example of a scalable data service is as follows: In addition to the failover resource group that contains the network resources, a scalable data service requires a separate resource group (*scalable resource group*) for the application resources.

The Oracle application has two components, a server and a listener. Sun supplies the Sun Cluster HA for Oracle data service, and therefore these components have already been mapped into Sun Cluster resource types. Both of these resource types are associated with resources and resource groups.

Because this example is a failover data service, the example uses logical hostname network resources, which are the IP addresses that fail over from a primary node to a secondary node. Place the logical hostname resources into a failover resource group, and then place the Oracle server resources and listener resources into the same resource group. This ordering enables all of the resources to fail over as a group.

For Sun Cluster HA for Oracle to run on the cluster, you must define the following objects.

- `LogicalHostname` resource type – This resource type is built in, and therefore you do not need to explicitly register the resource type.
- Oracle resource types – Sun Cluster HA for Oracle defines two Oracle resource types—a database server and a listener.
- Logical hostname resources – These resources host the IP addresses that fail over in a node failure.
- Oracle resources – You must specify two resource instances for Sun Cluster HA for Oracle — a server and a listener.
- Failover resource group – This container is composed of the Oracle server and listener and logical hostname resources that will fail over as a group.

Tools for Data Service Resource Administration

This section describes the tools that you can use to perform installation and configuration tasks.

SunPlex Manager Graphical User Interface (GUI)

SunPlex Manager is a web-based tool that enables you to perform the following tasks.

- Installing a cluster
- Administering a cluster
- Creating and configuring resources and resource groups
- Configuring data services with the Sun Cluster software

Sun Cluster Manager provides wizards to automate the configuration of Sun Cluster data services for the following applications.

- Apache Web Server
- NFS
- Oracle
- Oracle Real Application Clusters
- SAP Web Application Server

Each wizard enables you to configure Sun Cluster resources that the data service requires. The wizard does not automate the installation and configuration of the application software to run in a Sun Cluster configuration. To install and configure application software to run in a Sun Cluster configuration, use utilities of the application and Sun Cluster maintenance commands. For more information, see your application documentation and the Sun Cluster documentation set. Each wizard supports only a limited subset of configuration options for a data service. To configure options that a wizard does not support, use Sun Cluster Manager or Sun Cluster maintenance commands to configure the data service manually. For more information, see the Sun Cluster documentation.

Sun Cluster Manager provides wizards to automate the configuration of the following Sun Cluster resources.

- Logical hostname resource
- Shared address resource
- Highly available storage resource

You can use a resource that you create by using a wizard with any data service regardless of how you configure the data service.

For instructions for using use SunPlex Manager to install cluster software, see *Sun Cluster Software Installation Guide for Solaris OS*. SunPlex Manager provides online help for most administrative tasks.

SPARC: The Sun Cluster Module for the Sun Management Center GUI

The Sun Cluster module enables you to monitor clusters and to perform some operations on resources and resource groups from the Sun Management Center GUI. See the *Sun Cluster Software Installation Guide for Solaris OS* for information about installation requirements and procedures for the Sun Cluster module. Go to `http://docs.sun.com` to access the Sun Management Center software documentation set, which provides additional information about Sun Management Center.

`clsetup` Utility

The `clsetup`(1CL) utility is a menu-driven interface that you can use for general Sun Cluster administration. You can also use this utility to configure data service resources and resource groups. Select option 2 from the `clsetup` main menu to launch the Resource Group Manager submenu.

Sun Cluster Maintenance Commands

You can use the Sun Cluster maintenance commands to register and configure data service resources. See the procedure for how to register and configure your data service in the book for the data service. If, for example, you are using Sun Cluster HA for Oracle, see "Registering and Configuring Sun Cluster HA for Oracle" in *Sun Cluster Data Service for Oracle Guide for Solaris OS*.

For more information about how to use the commands to administer data service resources, see Chapter 2, "Administering Data Service Resources."

Summary by Task of Tools for Administering Data Service Resources

The following table summarizes by task which tool you can use to administer data service resources. For more information about these tasks and for details about how to use the command line to complete related procedures, see Chapter 2, "Administering Data Service Resources."

TABLE 1–2 Tools for Administering Data Service Resources

Task	SunPlex Manager	SPARC: Sun Management Center	`clsetup` Utility
Register a resource type	+	–	+
Create a resource group	+	–	+
Add a resource to a resource group	+	–	+
Suspend the automatic recovery actions of a resource group	+	–	+
Resume the automatic recovery actions of a resource group	+	–	+
Bring a resource group online	+	+	+
Remove a resource group	+	+	+
Remove a resource	+	+	+
Switch the current primary of a resource group	+	–	+
Enable a resource	+	+	+
Disable a resource	+	+	+
Move a resource group to the unmanaged state	+	–	+
Display resource type, resource group, and resource configuration information	+	+	+
Change resource properties	+	–	+
Clear the `STOP_FAILED` error flag on resources	+	–	+
Clear the `START_FAILED` resource state for a resource	+	–	+
Add a node to a resource group	+	–	+

◆ ◆ ◆ CHAPTER 2

Administering Data Service Resources

This chapter describes how to use the Sun Cluster maintenance commands to manage resources, resource groups, and resource types within the cluster. To determine if you can use other tools to complete a procedure, see “Tools for Data Service Resource Administration” on page 22.

For overview information about resource types, resource groups, and resources, see Chapter 1, “Planning for Sun Cluster Data Services,” and *Sun Cluster Concepts Guide for Solaris OS*.

This chapter contains the following sections.

- “Clearing the `STOP_FAILED` Error Flag on Resources” on page 82
- “Clearing the `Start_failed` Resource State” on page 84
- “Upgrading a Preregistered Resource Type” on page 89
- “Reregistering Preregistered Resource Types After Inadvertent Deletion” on page 91
- “Adding or Removing a Node to or From a Resource Group” on page 92
- “Migrating the Application From a Global-Cluster Voting Node to a Global-Cluster Non-Voting Node” on page 101
- “Synchronizing the Startups Between Resource Groups and Device Groups” on page 104
- “Configuring an `HAStoragePlus` Resource for Cluster File Systems” on page 108
- “Enabling Highly Available Local File Systems” on page 110
- “Upgrading From `HAStorage` to `HAStoragePlus`” on page 120
- “Modifying Online the Resource for a Highly Available File System” on page 124
- “Changing the Cluster File System to a Local File System in an `HAStoragePlus` Resource ” on page 133
- “Upgrading the `HAStoragePlus` Resource Type” on page 134
- “Distributing Online Resource Groups Among Cluster Nodes” on page 135
- “Replicating and Upgrading Configuration Data for Resource Groups, Resource Types, and Resources” on page 144
- “Enabling Solaris SMF Services to Run With Sun Cluster” on page 147
- “Tuning Fault Monitors for Sun Cluster Data Services” on page 158

Overview of Tasks for Administering Data Service Resources

The following table summarizes the tasks for installing and configuring Sun Cluster data services. The table also provides cross-references to detailed instructions for performing the tasks.

TABLE 2–1 Tasks for Administering Data Service Resources

Task	Instructions
Register a resource type	“How to Register a Resource Type” on page 32
Upgrade a resource type	“How to Migrate Existing Resources to a New Version of the Resource Type” on page 35 “How to Install and Register an Upgrade of a Resource Type” on page 34
Downgrade a resource type	“How to Downgrade a Resource to an Older Version of Its Resource Type” on page 40
Create failover or scalable resource groups	“How to Create a Failover Resource Group” on page 41 “How to Create a Scalable Resource Group” on page 43

TABLE 2–1 Tasks for Administering Data Service Resources *(Continued)*

Task	Instructions
Add logical hostnames or shared addresses and data service resources to resource groups	"How to Add a Logical Hostname Resource to a Resource Group by Using the `clsetup` Utility" on page 46 "How to Add a Logical Hostname Resource to a Resource Group Using the Command-Line Interface" on page 49 "How to Add a Shared Address Resource to a Resource Group by Using the `clsetup` Utility" on page 51 "How to Add a Shared Address Resource to a Resource Group Using the Command-Line Interface" on page 53 "How to Add a Failover Application Resource to a Resource Group" on page 55 "How to Add a Scalable Application Resource to a Resource Group" on page 58
Enable resources and resource monitors, manage the resource group, and bring the resource group and its associated resources online	"How to Enable a Resource" on page 62 "How to Bring Online Resource Groups" on page 61
Quiesce a resource group	"How to Quiesce a Resource Group" on page 63 "How to Quiesce a Resource Group Immediately" on page 64
Suspend and resume automatic recovery actions of a resource group	"How to Suspend the Automatic Recovery Actions of a Resource Group" on page 66 "How to Suspend the Automatic Recovery Actions of a Resource Group Immediately" on page 66 "How to Resume the Automatic Recovery Actions of a Resource Group" on page 66
Disable and enable resource monitors independent of the resource	"How to Disable a Resource Fault Monitor" on page 67 "How to Enable a Resource Fault Monitor" on page 68
Remove resource types from the cluster	"How to Remove a Resource Type" on page 69
Remove resource groups from the cluster	"How to Remove a Resource Group" on page 70
Remove resources from resource groups	"How to Remove a Resource" on page 71
Switch the primary for a resource group	"How to Switch the Current Primary of a Resource Group" on page 72
Disable resources and move their resource group into the `UNMANAGED` state	"How to Disable a Resource and Move Its Resource Group Into the `UNMANAGED` State" on page 75

TABLE 2–1 Tasks for Administering Data Service Resources *(Continued)*

Task	Instructions
Display resource type, resource group, and resource configuration information	"Displaying Resource Type, Resource Group, and Resource Configuration Information" on page 77
Change resource type, resource group, and resource properties	"How to Change Resource Type Properties" on page 78 "How to Change Resource Group Properties" on page 79 "How to Change Resource Properties" on page 80
Clear error flags for failed Resource Group Manager (RGM) processes	"How to Clear the `STOP_FAILED` Error Flag on Resources" on page 83
Clear the `Start_failed` resource state	"How to Clear a `Start_failed` Resource State by Switching Over a Resource Group" on page 84 "How to Clear a `Start_failed` Resource State by Restarting a Resource Group" on page 86 "How to Clear a `Start_failed` Resource State by Disabling and Enabling a Resource" on page 88
Reregister the built-in resource types `LogicalHostname` and `SharedAddress`	"How to Reregister Preregistered Resource Types After Inadvertent Deletion" on page 91
Update the network interface ID list for the network resources, and update the node list for the resource group	"Adding a Node to a Resource Group" on page 93
Remove a node from a resource group	"Removing a Node From a Resource Group" on page 96
Migrate an application from a global-cluster voting node to a global-cluster non-voting node	"How to Migrate the Application From a Global-Cluster Voting Node to a Global-Cluster Non-Voting Node" on page 101
Set up `HAStoragePlus` for resource groups to synchronize the startups between those resource groups and device groups	"How to Set Up the `HAStoragePlus` Resource Type for New Resources" on page 105 "How to Set Up the `HAStoragePlus` Resource Type for Existing Resources" on page 108 "How to Set Up the `HAStoragePlus` Resource for Cluster File Systems" on page 109 "How to Set Up the `HAStoragePlus` Resource Type by Using the `clsetup` Utility" on page 113
Set up the `HAStoragePlus` to make a local Solaris ZFS highly available	"How to Set Up the `HAStoragePlus` Resource Type to Make a Local Solaris ZFS Highly Available" on page 117
Modify online the resource for a highly available file system	"Modifying Online the Resource for a Highly Available File System" on page 124

TABLE 2–1 Tasks for Administering Data Service Resources *(Continued)*

Task	Instructions
Change the cluster file system to local file system in an `HAStoragePlus` resource	"Changing the Cluster File System to a Local File System in an `HAStoragePlus` Resource " on page 133
Upgrade the built-in resource types `LogicalHostname` and `SharedAddress`	"Upgrading a Resource Type" on page 33 "Upgrading a Preregistered Resource Type" on page 89
Upgrade the `HAStoragePlus` resource type	"Upgrading a Resource Type" on page 33 "Upgrading the `HAStoragePlus` Resource Type" on page 134
Distribute online resource groups among cluster nodes	"Distributing Online Resource Groups Among Cluster Nodes" on page 135
Replicate and upgrade configuration data for resource groups, resource types, and resources	"Replicating and Upgrading Configuration Data for Resource Groups, Resource Types, and Resources" on page 144
Enable Solaris SMF services to run with Sun Cluster	"Enabling Solaris SMF Services to Run With Sun Cluster" on page 147
Tune fault monitors for Sun Cluster data services	"Tuning Fault Monitors for Sun Cluster Data Services" on page 158

Note – The procedures in this chapter describe how to use the Sun Cluster maintenance commands to complete these tasks. Other tools also enable you to administer your resources. See "Tools for Data Service Resource Administration" on page 22 for details about these options.

Configuring and Administering Sun Cluster Data Services

Configuring a Sun Cluster data service involves the following tasks.

- Registering a resource type
- Upgrading a resource type
- Creating resource groups
- Adding resources to resource groups
- Bringing online resources

Use the procedures in this chapter to update your data service configuration after the initial configuration. For example, to change resource type, resource group, and resource properties, go to "Changing Resource Type, Resource Group, and Resource Properties" on page 77.

Registering a Resource Type

A resource type provides specification of common properties and callback methods that apply to all of the resources of the given type. You must register a resource type before you create a resource of that type. For details about resource types, see Chapter 1, "Planning for Sun Cluster Data Services."

An administrator can register a resource type for a zone cluster by specifying a resource type registration (RTR) file that resides inside the zone cluster. In other words, the file must be under the zone root path. The RTR file inside the zone cluster cannot have the `Global_zone` property set to `TRUE`. The RTR file inside the zone cluster cannot be of type `RTR_LOGICAL_HOSTNAME` or `RTR_SHARED_ADDRESS`.

The administrator can also register a resource type for a zone cluster from the location `/usr/cluster/lib/rgm/rtreg`. The administrator in the zone cluster cannot modify any RTR files in this directory. This enables registering system resource types for a zone cluster, even when the RTR file has one of the properties that cannot be set directly from the zone cluster. This process provides a secure way of delivering system resource types.

The resource types in the `/usr/cluster/lib/rgm/rtreg` directory are for the exclusive use of the global cluster.

▼ How to Register a Resource Type

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the name for the resource type that you plan to register. The resource type name is an abbreviation for the data service name. For information about resource type names of data services that are supplied with Sun Cluster, see the release notes for your release of Sun Cluster.

1. **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2. **Register the resource type.**

   ```
   # clresourcetype register resource-type
   ```

resource-type	Specifies name of the resource type to add. See the release notes for your release of Sun Cluster to determine the predefined name to supply.

3. **Verify that the resource type has been registered.**

   ```
   # clresourcetype show
   ```

Example 2–1 Registering a Resource Type

The following example registers the SUNW.krb5 resource type, which represents the Sun Java System Web Server application in a Sun Cluster configuration.

```
# clresourcetype register SUNW.krb5
# clresourcetype show SUNW.krb5

Resource Type:                                  SUNW.krb5
RT_description:                                  HA-Kerberos KDC server for Sun Cluster
RT_version:                                      3.2
API_version:                                     6
RT_basedir:                                      /opt/SUNWsckrb5/bin
Single_instance:                                 False
Proxy:                                           False
Init_nodes:                                      All potential masters
Installed_nodes:                                 <All>
Failover:                                        True
Pkglist:                                         SUNWsckrb5
RT_system:                                       False
```

Next Steps After registering resource types, you can create resource groups and add resources to the resource group. For details, see "Creating a Resource Group" on page 41.

See Also The following man pages:

- `clresourcetype(1CL)`
- `clresourcegroup(1CL)`
- `clresource(1CL)`

Upgrading a Resource Type

Upgrading a resource type enables you to use new features that are introduced in the new version of the resource type. A new version of a resource type might differ from a previous version in the following ways.

- Default settings of resource type properties might change.
- New extension properties of the resource type might be introduced.
- Existing extension properties of the resource type might be withdrawn.
- The set of standard properties that are declared for the resource type might change.
- The attributes of resource properties such as `min`, `max`, `arraymin`, `arraymax`, `default`, and `tunability` might change.

- The set of declared methods might differ.
- The implementation of methods or the fault monitor might change.

Upgrading a resource type involves the tasks that are explained in the following sections:

1. "How to Install and Register an Upgrade of a Resource Type" on page 34
2. "How to Migrate Existing Resources to a New Version of the Resource Type" on page 35

▼ How to Install and Register an Upgrade of a Resource Type

The instructions that follow explain how to use the `clresource(1CL)` command to perform this task. However, you are not restricted to using the `clresource` command for this task. Instead of the `clresource` command, you can use SunPlex Manager or the Resource Group option of the `clsetup(1CL)` command to perform this task.

Before You Begin Consult the documentation for the resource type to determine what you must do before installing the upgrade package on a node. One action from the following list will be required:

- You must reboot the node in noncluster mode.
- You may leave the node running in cluster mode, but you must turn off monitoring of all instances of the resource type.
- You may leave the node running in cluster mode and leave monitoring turned on for all instances of the resource type.

If you must reboot the node in noncluster mode, prevent a loss of service by performing a rolling upgrade. In a rolling upgrade, you install the package on each node individually while leaving the remaining nodes running in cluster mode.

1 **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Install the package for the resource type upgrade on all cluster nodes where instances of the resource type are to be brought online.**

3 **Register the new version of the resource type.**

To ensure that the correct version of the resource type is registered, you must specify the following information:

- The resource type name

- The resource type registration (RTR) file that defines the resource type

```
# clresourcetype register -f path-to-new-rtr-file  resource-type-name
```

The format of the resource type name is as follows:

vendor-id.*base-rt-name*:*rt-version*

For an explanation of this format, see "Format of Resource Type Names" on page 224.

4 **Display the newly registered resource type.**

```
# clresourcetype show  resource-type-name
```

5 **If necessary, set the `Installed_nodes` property to the nodes where the package for the resource type upgrade is installed.**

You must perform this step if the package for the resource type upgrade is not installed on all cluster nodes.

The `nodelist` property of all resource groups that contain instances of the resource type must be a subset of the `Installed_nodes` property of the resource type.

```
# clresourcetype set -n  installed-node-list  resource-type
```

`-n` *installed-node-list*	Specifies the names of nodes on which this resource type is installed.

▼ How to Migrate Existing Resources to a New Version of the Resource Type

The instructions that follow explain how to use the `clresource(1CL)` command to perform this task. However, you are not restricted to using the `clresource` command for this task. Instead of the `clresource` command, you can use SunPlex Manager or the Resource Group option of the `clsetup(1CL)` command to perform this task.

Before You Begin Consult the instructions for upgrading the resource type to determine when you can migrate resources to a new version of the resource type.

- Any time
- Only when the resource is unmonitored
- Only when the resource is offline
- Only when the resource is disabled
- Only when the resource group is unmanaged

The instructions might state that you cannot upgrade your existing version of the resource. If you cannot migrate the resource, consider the following alternatives:

- Deleting the resource and replacing it with a new resource of the upgraded version
- Leaving the resource at the old version of the resource type

1. **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2. **For each resource of the resource type that is to be migrated, change the state of the resource or its resource group to the appropriate state.**

   - **If you can migrate the resource at any time, no action is required.**
   - **If you can migrate the resource only when the resource is unmonitored, type the following command:**

     ```
     # clresource unmonitor resource
     ```

   - **If you can migrate the resource only when the resource is offline, type the following command:**

     ```
     # clresource disable resource
     ```

     Note – If other resources depend on the resource that you are migrating, this step fails. In this situation, consult the error message that is printed to determine the names of the dependent resources. Then repeat this step, specifying a comma-separated list that contains the resource that you are migrating and any dependent resources.

   - **If you can migrate the resource only when the resource is disabled, type the following command:**

     ```
     # clresource disable resource
     ```

     Note – If other resources depend on the resource that you are migrating, this step fails. In this situation, consult the error message that is printed to determine the names of the dependent resources. Then repeat this step, specifying a comma-separated list that contains the resource that you are migrating and any dependent resources.

   - **If you can migrate the resource only when the resource group is unmanaged, type the following commands:**

     ```
     # clresource disable -g resource-group +
     # clresourcegroup offline resource-group
     # clresourcegroup unmanage resource-group
     ```

     The replaceable items in these commands are as follows:

resource-group	Specifies the resource group that is to be unmanaged

3 **For each resource of the resource type that is to be migrated, change the `Type_version` property to the new version.**

If necessary, set other properties of the same resource to appropriate values in the same command. To set these properties, specify the `-p` option in the command.

To determine whether you are required to set other properties, consult the instructions for upgrading the resource type. You might be required to set other properties for the following reasons:

- An extension property has been introduced in the new version of the resource type.
- The default value of an existing property has been changed in the new version of the resource type.

```
# clresource set -p Type_version=new-version \
[-p extension-property=new-value] [-p standard-property=new-value] resource
```

Note – If the existing version of the resource type does not support upgrades to the new version, this step fails.

---

4 **Restore the previous state of the resource or resource group by reversing the command that you typed in Step 2.**

- **If you can migrate the resource at any time, no action is required.**

  Note – After migrating a resource that can be migrated at any time, the resource probe might not display the correct resource type version. In this situation, disable and re-enable the resource's fault monitor to ensure that the resource probe displays the correct resource type version.

- **If you can migrate the resource only when the resource is unmonitored, type the following command:**

  ```
  # clresource monitor resource
  ```

- **If you can migrate the resource only when the resource is offline, type the following command:**

  ```
  # clresource enable resource
  ```

  Note – If you disabled in Step 2 other resources that depend on the resource that you are migrating, enable the dependent resources also.

- **If you can migrate the resource only when the resource is disabled, type the following command:**

  ```
  # clresource enable resource
  ```

  Note – If you disabled in Step 2 other resources that depend on the resource that you are migrating, enable the dependent resources also.

- **If you can migrate the resource only when the resource group is unmanaged, type the following commands:**

  ```
  # clresource enable -g resource-group +
  # clresourcegroup manage resource-group
  # clresourcegroup online resource-group
  ```

Example 2–2 Migrating a Resource That Can Be Migrated Only When Offline

This example shows the migration of a resource that can be migrated only when the resource is offline. The new resource type package contains methods that are located in new paths. Because the methods are not overwritten during the installation, the resource does not need to be disabled until after the upgraded resource type is installed.

The characteristics of the resource in this example are as follows:

- The new resource type version is 2.0.
- The resource name is `myresource`.
- The resource type name is `myrt`.
- The new RTR file is in `/opt/XYZmyrt/etc/XYZ.myrt`.
- No dependencies on the resource that is to be migrated exist.
- The resource that is to be migrated can be taken offline while leaving the containing resource group online.

This example assumes that the upgrade package is already installed on all cluster nodes according to the supplier's directions.

```
# clresourcetype register -f /opt/XYZmyrt/etc/XYZ.myrt myrt
# clresource disable myresource
# clresource set -p Type_version=2.0 myresource
# clresource enable myresource
```

Example 2–3 Migrating a Resource That Can Be Migrated Only When Unmonitored

This example shows the migration of a resource that can be migrated only when the resource is unmonitored. The new resource type package contains only the monitor and RTR file. Because the monitor is overwritten during installation, monitoring of the resource must be disabled before the upgrade package is installed.

The characteristics of the resource in this example are as follows:

- The new resource type version is 2.0.
- The resource name is `myresource`.
- The resource type name is `myrt`.
- The new RTR file is in `/opt/XYZmyrt/etc/XYZ.myrt`.

The following operations are performed in this example.

1. Before the upgrade package is installed, the following command is run to disable monitoring of the resource:

   ```
   # clresource unmonitor  myresource
   ```
2. The upgrade package is installed on all cluster nodes according to the supplier's directions.
3. To register the new version of the resource type, the following command is run:

   ```
   # clresourcetype register -f /opt/XYZmyrt/etc/XYZ.myrt myrt
   ```
4. To change the `Type_version` property to the new version, the following command is run:

   ```
   # clresource set -p Type_version=2.0 myresource
   ```
5. To enable monitoring of the resource after its migration, the following command is run:

   ```
   # clresource monitor myresource
   ```

Downgrading a Resource Type

You can downgrade a resource to an older version of its resource type. The conditions for downgrading a resource to an older version of the resource type are more restrictive than the conditions for upgrading to a newer version of the resource type. The resource group that contains the resource must be unmanaged.

▼ How to Downgrade a Resource to an Older Version of Its Resource Type

The instructions that follow explain how to use the `clresource(1CL)` command to perform this task. However, you are not restricted to using the `clresource` command for this task. Instead of the `clresource` command, you can use SunPlex Manager or the Resource Group option of the `clsetup(1CL)` command to perform this task.

1. **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` and `solaris.cluster.admin` RBAC authorizations.**

2. **Switch offline the resource group that contains the resource that you are downgrading.**

   ```
   clresourcegroup offline resource-group
   ```

3. **Disable all resources in the resource group that contains the resource that you are downgrading.**

   ```
   clresource disable -g resource-group +
   ```

4. **Unmanage the resource group that contains the resource that you are downgrading.**

   ```
   clresourcegroup unmanage resource-group
   ```

5. **If necessary, reregister the old version of the resource type to which you are downgrading.**

   Perform this step only if the version to which you are downgrading is no longer registered. If the version to which you are downgrading is still registered, omit this step.

   ```
   clresourcetype register resource-type-name
   ```

6. **For the resource that you are downgrading, set the `Type_version` property to old version to which you are downgrading.**

   If necessary, edit other properties of the same resource to appropriate values in the same command.

   ```
   clresource set -p Type_version=old-version  resource-todowngrade
   ```

7. **Enable all the resources that you disabled in Step 3.**

   ```
   # clresource enable -g resource-group +
   ```

8. **Bring to a managed state the resource group that contains the resource that you downgraded.**

   ```
   # clresourcegroup manage resource-group
   ```

9. **Bring online the resource group that contains the resource that you downgraded.**

   ```
   # clresourcegroup online resource-group
   ```

Creating a Resource Group

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 must create an empty resource group before you place resources into it. A resource group can be configured to run in global-cluster non-voting nodes.

Note – The global-cluster non voting nodes that are specified in the resource group's node list do not need to exist when the resource group is created. If the node specified in the node list is not detected by the RGM, a warning message is displayed but does not result in an error.

The two resource group types are **failover** and **scalable**. A failover resource group can be online on one node only at any time, while a scalable resource group can be online on multiple nodes simultaneously.

The following procedures explain how to use the `clresourcegroup(1CL)` command to create a resource group.

For conceptual information about resource groups, see Chapter 1, "Planning for Sun Cluster Data Services," and *Sun Cluster Concepts Guide for Solaris OS*.

▼ How to Create a Failover Resource Group

A failover resource group contains the following types of resources:

- Network address resources, which are instances of the built-in resource types `LogicalHostname` and `SharedAddress`
- Failover resources, which are the data service application resources for a failover data service

The network address resources and their dependent data service resources move between cluster nodes when data services fail over or are switched over.

Note – Perform this procedure from any cluster node.

---

1 **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **Create the failover resource group.**

```
# clresourcegroup create [-n node-zone-list] resource-group
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes that can master this resource group. The format of each entry in the list is *node*`:`*zone*. In this

format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*.

This list is optional. If you omit this list, the resource group is created on all nodes in the cluster.

Note – To achieve highest availability, specify global-cluster non-voting nodes on different global-cluster voting nodes in a failover resource group's node list instead of different nodes on the same global-cluster voting node.

---

resource-group Specifies your choice of the name of the failover resource group to add. This name must begin with an ASCII character.

3 **Verify that the resource group has been created.**

```
# clresourcegroup show resource-group
```

Example 2–4 Creating a Failover Resource Group

This example shows the creation of the failover resource group `resource-group-1`. The global cluster voting nodes `phys-schost-1` and `phys-schost-2` can master this resource group.

```
# clresourcegroup create -n phys-schost1,phys-schost-2 resource-group-1
# clresourcegroup show -v resource-group-1

=== Resource Groups and Resources ===

Resource Group:                                 resource-group1
RG_description:                                 <NULL>
RG_mode:                                        Failover
RG_state:                                       Unmanaged
RG_project_name:                                default
RG_affinities:                                  <NULL>
RG_SLM_type:                                    manual
Auto_start_on_new_cluster:                      True
Failback:                                       False
Nodelist:                                       phys-schost-1 phys-schost-2
Maximum_primaries:                              1
Desired_primaries:                              1
RG_dependencies:                                <NULL>
Implicit_network_dependencies:                  True
Global_resources_used:                          <All>
```

```
Pingpong_interval:                               3600
Pathprefix:                                      <NULL>
RG_System:                                       False
Suspend_automatic_recovery:                      False
```

Next Steps After you create a failover resource group, you can add application resources to this resource group. See "Tools for Adding Resources to Resource Groups" on page 45 for the procedure.

See Also The `clresourcegroup(1CL)` man page.

▼ How to Create a Scalable Resource Group

A scalable resource group is used with scalable services. The shared address feature is the Sun Cluster networking facility that enables the multiple instances of a scalable service to appear as a single service. You must first create a failover resource group that contains the shared addresses on which the scalable resources depend. Next, create a scalable resource group, and add scalable resources to that group. The node list of a scalable resource group or of the shared address resource group must not contain more than one global-cluster non-voting node on the same node. Each instance of the scalable service must run on a different cluster node.

You can configure a scalable resource group to run in a global-cluster non-voting node as well. Do not configure a scalable resource to run in multiple global-cluster non-voting nodes on the same node.

Note – Perform this procedure from any cluster node.

---

1. **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2. **Create the failover resource group that holds the shared addresses that the scalable resource is to use.**

3. **Create the scalable resource group.**

```
# clresourcegroup create\-p Maximum_primaries=m\-p Desired_primaries=n\
-p RG_dependencies=depend-resource-group\
[-n node-zone-list] resource-group
```

`-p Maximum_primaries=`*m*	Specifies the maximum number of active primaries for this resource group.

`-p Desired_primaries=`*n*	Specifies the number of active primaries on which the resource group should attempt to start.
`-p RG_dependencies=`*depend-resource-group*	Identifies the resource group that contains the shared address resource on which the resource group that is being created depends.
`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes in which this resource group is to be available. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the resource group is created on all nodes in the cluster. The node list of the scalable resource can contain the same list or a subset of *nodename:zonename* pairs as the node list of the shared address resource
resource-group	Specifies your choice of the name of the scalable resource group to add. This name must begin with an ASCII character.

4 **Verify that the scalable resource group has been created.**

```
# clresourcegroup show resource-group
```

Example 2–5 Creating a Scalable Resource Group

This example shows the creation of the scalable resource group `resource-group-1`. This resource group is to be hosted in the global cluster of nodes `phys-schost-1` and `phys-schost-2`. The scalable resource group depends on the failover resource group `resource-group-2`, which contains the shared address resources.

```
# clresourcegroup create\
-p Maximum_primaries=2\
-p Desired_primaries=2\
-p RG_dependencies=resource-group-2\
-n phys-schost-1, phys-schost-2\
resource-group-1

# clresourcegroup show resource-group-1

=== Resource Groups and Resources ===

Resource Group:                                  resource-group-1
RG_description:                                  <NULL>
RG_mode:                                         Scalable
RG_state:                                        Unmanaged
RG_project_name:                                 default
RG_affinities:                                   <NULL>
Auto_start_on_new_cluster:                       True
Failback:                                        False
Nodelist:                                        phys-schost-1 phys-schost-2
Maximum_primaries:                               2
Desired_primaries:                               2
RG_dependencies:                                 resource-group2
Implicit_network_dependencies:                   True
Global_resources_used:                           <All>
Pingpong_interval:                               3600
Pathprefix:                                      <NULL>
RG_System:                                       False
Suspend_automatic_recovery:                      False
```

Next Steps After you have created a scalable resource group, you can add scalable application resources to the resource group. See "How to Add a Scalable Application Resource to a Resource Group" on page 58 for details.

See Also The clresourcegroup(1CL) man page.

Tools for Adding Resources to Resource Groups

A resource is an instantiation of a resource type. You must add resources to a resource group before the RGM can manage the resources. This section describes the following three resource types.

- Logical hostname resources
- Shared-address resources
- Data service (application) resources

Sun Cluster provides the following tools for adding resources to resource groups:

- **Sun Cluster Manager.** For more information, see the Sun Cluster Manager online help.
- **The `clsetup(1CL)` utility.**
- **Sun Cluster maintenance commands.**

You can use the wizards in the Sun Cluster Manager, the `clsetup` utility, or the Sun cluster maintenance commands to add the logical hostname resources and shared-address resources to the resource group.

Sun Cluster Manager and the `clsetup` utility enable you to add resources to the resource group interactively. Configuring these resources interactively reduces the possibility for configuration errors that might result from command syntax errors or omissions. Sun Cluster Manager and the `clsetup` utility ensure that all required resources are created and that all required dependencies between resources are set.

Always add logical hostname resources and shared address resources to failover resource groups. Add data service resources for failover data services to failover resource groups. Failover resource groups contain both the logical hostname resources and the application resources for the data service. Scalable resource groups contain only the application resources for scalable services. The shared address resources on which the scalable service depends must reside in a separate failover resource group. You must specify dependencies between the scalable application resources and the shared address resources for the data service to scale across cluster nodes.

Note – The `DEPRECATED` flag marks the logical hostname or shared address resource as a deprecated address. These addresses are not suitable for outbound requests since they can migrate to a different cluster node due to a failover or switchover.

For more information about resources, see *Sun Cluster Concepts Guide for Solaris OS* and Chapter 1, "Planning for Sun Cluster Data Services."

▼ How to Add a Logical Hostname Resource to a Resource Group by Using the `clsetup` Utility

The following instructions explain how to add a logical hostname resource to a resource group by using the `clsetup` utility. Perform this procedure from one node only.

This procedure provides the long forms of the Sun Cluster maintenance commands. Most commands also have short forms. Except for the forms of the command names, the commands are identical. For a list of the commands and their short forms, see Appendix A, "Sun Cluster Object-Oriented Commands."

Before You Begin Ensure that the following prerequisites are met:

- An entry for each logical hostname that is to be made available by the resource is added to the name service database.
- If you are using IP Networking Multipathing (IPMP) groups, the groups are configured on the nodes where the logical hostname resource can be brought online.
- Any global-cluster non-voting node that can master the resource is already configured on your cluster nodes.

Ensure that you have the following information:

- The hostnames that you plan to add to the resource group

1 **Become superuser on any cluster node.**

2 **Start the `clsetup` utility.**

```
# clsetup
```

The `clsetup` main menu is displayed.

3 **Type the number that corresponds to the option for data services and press Return.**

The Data Services menu is displayed.

4 **Type the number that corresponds to the option for configuring the Logical Hostname resource and press Return.**

The `clsetup` utility displays the list of prerequisites for performing this task.

5 **Verify that the prerequisites are met, and press Return to continue.**

The `clsetup` utility displays a list of the cluster nodes where the logical hostname resource can be brought online.

6 **Select the nodes where the logical hostname resource can be brought online.**

- **To accept the default selection of all listed nodes in an arbitrary order, type** a **and press Return.**

- **To select a subset of the listed nodes, type a comma-separated or space-separated list of the numbers that correspond to the nodes. Then press Return.**

- **To select all nodes in a particular order, type a comma-separated or space-separated ordered list of the numbers that correspond to the nodes and press Return.**

  Ensure that the nodes are listed in the order in which the nodes are to appear in the logical hostname resource group's node list. The first node in the list is the primary node of this resource group.

7 **To confirm your selection of nodes, type `d` and press Return.**

The `clsetup` utility displays a screen where you can specify the logical hostname that the resource is to make available.

8 **Type the logical hostname that this resource is to make available and press Return.**

The `clsetup` utility displays the names of the Sun Cluster objects that the utility will create.

9 **If you require a different name for any Sun Cluster object, change the name as follows.**

a. **Type the number that corresponds to the name that you are changing and press Return.**

The `clsetup` utility displays a screen where you can specify the new name.

b. **At the New Value prompt, type the new name and press Return.**

The `clsetup` utility returns you to the list of the names of the Sun Cluster objects that the utility will create.

10 **To confirm your selection of Sun Cluster object names, type `d` and press Return.**

The `clsetup` utility displays information about the Sun Cluster configuration that the utility will create.

11 **To create the configuration, type `c` and press Return.**

The `clsetup` utility displays a progress message to indicate that the utility is running commands to create the configuration. When configuration is complete, the `clsetup` utility displays the commands that the utility ran to create the configuration.

12 **(Optional) Type `q` and press Return repeatedly until you quit the `clsetup` utility.**

If you prefer, you can leave the `clsetup` utility running while you perform other required tasks before using the utility again. If you choose to quit `clsetup`, the utility recognizes your existing logical hostname resource group when you restart the utility.

13 **Verify that the logical hostname resource has been created.**

Use the `clresource`(1CL) utility for this purpose. By default, the `clsetup` utility assigns the name *node_name*-`rg` to the resource group.

```
# clresource show node_name-rg
```

▼ How to Add a Logical Hostname Resource to a Resource Group Using the Command-Line Interface

---

Note – When you add a logical hostname resource to a resource group, the extension properties of the resource are set to their default values. To specify a nondefault value, you must modify the resource after you add the resource to a resource group. For more information, see "How to Modify a Logical Hostname Resource or a Shared Address Resource" on page 82.

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the following information.

- The name of the failover resource group to which you are adding the resource
- The hostnames that you plan to add to the resource group

1 **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **Add the logical hostname resource to the resource group.**

```
# clreslogicalhostname create -g resource-group -h hostnamelist, ... [-N netiflist] resource
```

`-g` *resource-group*	Specifies the name of the resource group in which this resource resides.
`-h` *hostnamelist, …*	Specifies a comma-separated list of UNIX hostnames (logical hostnames) by which clients communicate with services in the resource group. When a logical hostname resource is added to a resource group that runs in a global-cluster non-voting node, the corresponding IP addresses are configured in that node. These IP addresses are available only to applications that are running in that global-cluster non-voting node. You must specify the fully qualified name with the `-h` option if you require a fully qualified hostname.
`-N` *netiflist*	Specifies an optional, comma-separated list that identifies the IPMP groups that are on each node. Each element in *netiflist* must be in the form of `netif@node`. `netif` can be given as an IPMP group name, such as `sc_ipmp0`. The node can be identified by the node name or node ID, such as `sc_ipmp0@1` or `sc_ipmp@phys-schost-1`

Note – Sun Cluster does not support the use of the adapter name for `netif`.

---

resource Specifies an optional resource name of your choice. You cannot use the fully qualified name in the resource name.

3 **Verify that the logical hostname resource has been added.**

```
# clresource show resource
```

Example 2–6 Adding a Logical Hostname Resource to a Resource Group

This example shows the addition of logical hostname resource (`resource-1`) to a resource group (`resource-group-1`).

```
# clreslogicalhostname create -g resource-group-1 -h schost-1 resource-1
# clresource show resource-1

=== Resources ===

Resource:                                         resource-1
Type:                                             SUNW.LogicalHostname:2
Type_version:                                     2
Group:                                            resource-group-1
R_description:
Resource_project_name:                            default
Enabled{phats1}:                                  True
Enabled{phats2}:                                  True
Monitored{phats1}:                                True
Monitored{phats2}:                                True
```

Example 2–7 Adding Logical Hostname Resources That Identify IPMP Groups

This example shows the addition of the following logical host name resources to the resource group `nfs-fo-rg`:

- A resource that is named `cs23-rs`, which identifies the IPMP group `sc_ipmp0` on node 1 and node 2
- A resource that is named `cs24-rs`, which identifies the IPMP group `sc_ipmp1` on node 1 and node 2

```
# clreslogicalhostname create -g nfs-fo-rg -h cs23-rs -N sc_ipmp0@1,sc_ipmp0@2 cs23-rs
# clreslogicalhostname create -g nfs-fo-rg -h cs24-rs -N sc_ipmp1@1,sc_ipmp1@2 cs24-rs
```

Next Steps After you add logical hostname resources, see "How to Bring Online Resource Groups" on page 61 to bring the resources online.

Troubleshooting Adding a resource causes the Sun Cluster software to validate the resource. If the validation fails, the `clreslogicalhostname` command prints an error message and exits. To determine why the validation failed, check the `syslog` on each node for an error message. The message appears on the node that performed the validation, not necessarily the node on which you ran the `clreslogicalhostname` command.

See Also The `clreslogicalhostname`(1CL) man page.

▼ How to Add a Shared Address Resource to a Resource Group by Using the `clsetup` Utility

The following instructions explain how to add a shared address resource to a resource group by using the `clsetup` utility. Perform this procedure from any cluster node.

This procedure provides the long forms of the Sun Cluster maintenance commands. Most commands also have short forms. Except for the forms of the command names, the commands are identical. For a list of the commands and their short forms, see Appendix A, "Sun Cluster Object-Oriented Commands."

Before You Begin Ensure that the following prerequisites are met:

- The shared address that is to be made available by the resource has an entry in a name service database.
- If you are using IP Networking Multipathing (IPMP) groups, the groups are configured on the nodes where the shared address resource can be brought online.
- Any global-cluster non-voting node that can master the resource is already configured on your cluster nodes.

Ensure that you have the following information:

- The hostnames that you plan to add to the resource group.

1 **Become superuser on any cluster node.**

2 **Start the `clsetup` utility.**

```
# clsetup
```

The `clsetup` main menu is displayed.

3 **Type the number that corresponds to the option for data services and press Return.**

The Data Services menu is displayed.

4 **Type the number that corresponds to the option for configuring the shared address resource and press Return.**

The `clsetup` utility displays the list of prerequisites for performing this task.

5 **Verify that the prerequisites are met, and press Return to continue.**

The `clsetup` utility displays a list of the cluster nodes where the shared address resource can be brought online.

6 **Select the nodes where the shared address resource can be brought online.**

- **To accept the default selection of all listed nodes in an arbitrary order, type** `a` **and press Return.**
- **To select a subset of the listed nodes, type a comma-separated or space-separated list of the numbers that correspond to the nodes. Then press Return.**
- **To select all nodes in a particular order, type a comma-separated or space-separated ordered list of the numbers that correspond to the nodes and press Return.**

7 **To confirm your selection of nodes, type** `d` **and press Return.**

The `clsetup` utility displays a screen where you can specify the shared address that the resource is to make available.

8 **Type the shared address that this resource is to make available and press Return.**

The `clsetup` utility displays the names of the Sun Cluster objects that the utility will create.

9 **If you require a different name for any Sun Cluster object, change the name as follows.**

a. **Type the number that corresponds to the name that you are changing and press Return.**

The `clsetup` utility displays a screen where you can specify the new name.

b. **At the New Value prompt, type the new name and press Return.**

The `clsetup` utility returns you to the list of the names of the Sun Cluster objects that the utility will create.

10 **To confirm your selection of Sun Cluster object names, type** `d` **and press Return.**

The `clsetup` utility displays information about the Sun Cluster configuration that the utility will create.

11 To create the configuration, type `c` and Press Return.

The `clsetup` utility displays a progress message to indicate that the utility is running commands to create the configuration. When configuration is complete, the `clsetup` utility displays the commands that the utility ran to create the configuration.

12 (Optional) Type `q` and press Return repeatedly until you quit the `clsetup` utility.

If you prefer, you can leave the `clsetup` utility running while you perform other required tasks before using the utility again. If you choose to quit `clsetup`, the utility recognizes your existing shared address resource group when you restart the utility.

13 Verify that the shared address resource has been created.

Use the `clresource(1CL)` utility for this purpose. By default, the `clsetup` utility assigns the name *node_name*`-rg` to the resource group.

```
# clresource show node_name-rg
```

▼ How to Add a Shared Address Resource to a Resource Group Using the Command-Line Interface

---

Note – When you add a shared address resource to a resource group, the extension properties of the resource are set to their default values. To specify a nondefault value, you must modify the resource after you add the resource to a resource group. For more information, see "How to Modify a Logical Hostname Resource or a Shared Address Resource" on page 82.

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the following information.

- The name of the resource group into which you are adding the resource. This group must be a failover resource group that you created previously.
- The hostnames that you plan to add to the resource group.

1 On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.

2 Add the shared address resource to the resource group.

```
# clressharedaddress create -g resource-group -h hostnamelist, ... \
[-X auxnodelist] [-N netiflist] resource
```

`-g` *resource-group*	Specifies the resource group name. In the node list of a shared address resource, do not specify more than one global-cluster non-voting node on the same global-cluster voting node. Specify the same list of *nodename:zonename* pairs as the node list of the scalable resource group.
`-h` *hostnamelist*, …	Specifies a comma-separated list of shared address hostnames.
`-X` *auxnodelist*	Specifies a comma-separated list of node names or IDs that identify the cluster nodes that can host the shared address but never serve as primary if failover occurs. These nodes are mutually exclusive, with the nodes identified as potential masters in the resource group's node list. If no auxiliary node list is explicitly specified, the list defaults to the list of all cluster node names that are not included in the node list of the resource group that contains the shared address resource.

Note – To ensure that a scalable service runs in all global-cluster non-voting nodes that were created to master the service, the complete list of nodes must be included in the node list of the shared address resource group or the `auxnodelist` of the shared address resource. If all the nodes are listed in the node list, the `auxnodelist` can be omitted.

`-N` *netiflist*	Specifies an optional, comma-separated list that identifies the IPMP groups that are on each node. Each element in *netiflist* must be in the form of `netif@node`. `netif` can be given as an IPMP group name, such as `sc_ipmp0`. The node can be identified by the node name or node ID, such as `sc_ipmp0@1` or `sc_ipmp@phys-schost-1`.

Note – Sun Cluster does not support the use of the adapter name for `netif`.

resource	Specifies an optional resource name of your choice.

3 **Verify that the shared address resource has been added and validated.**

```
# clresource show resource
```

Example 2–8 Adding a Shared Address Resource to a Resource Group

This example shows the addition of a shared address resource (`resource-1`) to a resource group (`resource-group-1`).

```
# clressharedaddress create -g resource-group-1 -h schost-1 resource-1
# clresource show resource-1

=== Resources ===

  Resource:                                        resource-1
  Type:                                            SUNW.SharedAddress:2
  Type_version:                                    2
  Group:                                           resource-group-1
  R_description:
  Resource_project_name:                           default
  Enabled{phats1}:                                 False
  Enabled{phats2}:                                 False
  Monitored{phats1}:                               True
  Monitored{phats2}:                               True
```

Next Steps After you add a shared address resource, use the procedure "How to Bring Online Resource Groups" on page 61 to enable the resource.

Troubleshooting Adding a resource causes the Sun Cluster software to validate the resource. If the validation fails, the `clressharedaddress` command prints an error message and exits. To determine why the validation failed, check the `syslog` on each node for an error message. The message appears on the node that performed the validation, not necessarily the node on which you ran the `clressharedaddress` command.

See Also The `clressharedaddress`(1CL) man page.

▼ How to Add a Failover Application Resource to a Resource Group

A failover application resource is an application resource that uses logical hostnames that you previously created in a failover resource group.

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the following information.

- The name of the failover resource group to which you are adding the resource
- The name of the resource type for the resource

- The logical hostname resources that the application resource uses, which are the logical hostnames that you previously included in the same resource group

---

Note – This procedure also applies to proxy resources.

---

1 **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **Add a failover application resource to the resource group.**

```
# clresource create -g resource-group -t resource-type \
[-p "extension-property[{node-specifier}]"=value, ...] [-p standard-property=value, ...] resource
```

`-g` *resource-group*
Specifies the name of a failover resource group. This resource group must already exist.

`-t` *resource-type*
Specifies the name of the resource type for the resource.

`-p` "*extension-property*[**{***node-specifier***}**]"=*value*, …
Specifies a comma-separated list of extension properties that you are setting for the resource. The extension properties that you can set depend on the resource type. To determine which extension properties to set, see the documentation for the resource type.

node-specifier is an *optional* qualifier to the `-p` and `-x` options. This qualifier indicates that the extension property or properties on *only* the specified node or nodes are to be set when the resource is created. The specified extension properties on other nodes in the cluster are not set. If you do not include *node-specifier*, the specified extension properties on all nodes in the cluster are set. You can specify a node name or a node identifier for *node-specifier*. Examples of the syntax of *node-specifier* include the following:

```
-p "myprop{phys-schost-1}"
```

The braces ({}) indicate that you are setting the specified extension property on only node `phys-schost-1`. For most shells, the double quotation marks (“) are required.

You can also use the following syntax to set an extension property in two different global-cluster voting nodes on two different nodes:

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

Note – The extension property that you specify with *node-specifier* must be declared in the RTR file as a per-node property. See Appendix B, “Standard Properties,” for information about the `Per_node` resource property attribute.

-p *standard-property*=*value*, …
: Specifies a comma-separated list of standard properties that you are setting for the resource. The standard properties that you can set depend on the resource type. To determine which standard properties to set, see the documentation for the resource type and Appendix B, "Standard Properties."

resource
: Specifies your choice of the name of the resource to add.

The resource is created in the enabled state.

3 **Verify that the failover application resource has been added and validated.**

```
# clresource show resource
```

Example 2–9 Adding a Failover Application Resource to a Resource Group

This example shows the addition of a resource (`resource-1`) to a resource group (`resource-group-1`). The resource depends on logical hostname resources (`schost-1`, `schost-2`), which must reside in the same failover resource groups that you defined previously.

```
# clresource create -g resource-group-1 -t resource-type-1 \
-p Network_resources_used=schost-1,schost2  resource-1\
# clresource show resource-1

=== Resources ===

  Resource:                                         resource-1
  Type:                                             resource-type-1
  Type_version:
  Group:                                            resource-group-1
  R_description:
  Resource_project_name:                            default
  Enabled{phats1}:                                  False
  Enabled{phats2}:                                  False
  Monitored{phats1}:                                True
  Monitored{phats2}:                                True
```

Next Steps After you add a failover application resource, use the procedure "How to Bring Online Resource Groups" on page 61 to enable the resource.

Troubleshooting Adding a resource causes the Sun Cluster software to validate the resource. If the validation fails, the `clresource` command prints an error message and exits. To determine why the validation failed, check the `syslog` on each node for an error message. The message appears on the node that performed the validation, not necessarily the node on which you ran the `clresource` command.

See Also The clresource(1CL) man page.

▼ How to Add a Scalable Application Resource to a Resource Group

A scalable application resource is an application resource that uses shared-address resources. The shared-address resources are in a failover resource group.

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the following information.

- The name of the scalable resource group to which you are adding the resource
- The name of the resource type for the resource
- The shared address resources that the scalable service resource uses, which are the shared addresses that you previously included in a failover resource group

---

Note – This procedure also applies to proxy resources.

---

1. **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2. **Add a scalable application resource to the resource group.**

```
# clresource create -g resource-group -t resource-type \
-p Network_resources_used=network-resource[,network-resource...] \
-p Scalable=True
[-p "extension-property[{node-specifier}]"=value, ...] [-p standard-property=value, ...] resource
```

`-g` *resource-group*
    Specifies the name of a scalable service resource group that you previously created.

`-t` *resource-type*
    Specifies the name of the resource type for this resource.

`-p Network_resources_used=` *network-resource*[,*network-resource*...]
    Specifies the list of network resources (shared addresses) on which this resource depends.

`-p Scalable=True`
    Specifies that this resource is scalable.

`-p` "*extension-property*[**{***node-specifier***}**]"=*value*, …
    Specifies a comma-separated list of extension properties that you are setting for the resource. The extension properties that you can set depend on the resource type. To determine which extension properties to set, see the documentation for the resource type.

node-specifier is an *optional* qualifier to the -p and -x options. This qualifier indicates that the extension property or properties on *only* the specified node or nodes are to be set when the resource is created. The specified extension properties on other nodes in the cluster are not set. If you do not include *node-specifier*, the specified extension properties on all nodes in the cluster are set. You can specify a node name or a node identifier for *node-specifier*. Examples of the syntax of *node-specifier* include the following:

```
-p "myprop{phys-schost-1}"
```

The braces ({}) indicate that you are setting the specified extension property on only node phys-schost-1. For most shells, the double quotation marks (") are required.

You can also use the following syntax to set an extension property in two different global-cluster voting nodes on two different nodes:

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

Note – The extension property that you specify with *node-specifier* must be declared in the RTR file as a per-node property. See Appendix B, "Standard Properties," for information about the Per_node resource property attribute.

-p *standard-property*=*value*, …
Specifies a comma-separated list of standard properties that you are setting for the resource. The standard properties that you can set depend on the resource type. For scalable services, you typically set the Port_list, Load_balancing_weights, and Load_balancing_policy properties. To determine which standard properties to set, see the documentation for the resource type and Appendix B, "Standard Properties."

resource
Specifies your choice of the name of the resource to add.

The resource is created in the enabled state.

3 Verify that the scalable application resource has been added and validated.

```
# clresource show resource
```

Example 2–10 Adding a Scalable Application Resource to a Resource Group

This example shows the addition of a resource (resource-1) to a resource group (resource-group-1). Note that resource-group-1 depends on the failover resource group that contains the network addresses that are in use (schost-1 and schost-2 in the following example). The resource depends on shared address resources (schost-1, schost-2), which must reside in one or more failover resource groups that you defined previously.

```
# clresource create -g resource-group-1 -t resource-type-1 \
-p Network_resources_used=schost-1,schost-2 resource-1 \
-p Scalable=True
# clresource show resource-1

=== Resources ===

  Resource:                                         resource-1
  Type:                                             resource-type-1
  Type_version:
  Group:                                            resource-group-1
  R_description:
  Resource_project_name:                            default
  Enabled{phats1}:                                  False
  Enabled{phats2}:                                  False
  Monitored{phats1}:                                True
  Monitored{phats2}:                                True
```

Next Steps After you add a scalable application resource, follow the procedure "How to Bring Online Resource Groups" on page 61 to enable the resource.

Troubleshooting Adding a resource causes the Sun Cluster software to validate the resource. If the validation fails, the `clresource` command prints an error message and exits. To determine why the validation failed, check the `syslog` on each node for an error message. The message appears on the node that performed the validation, not necessarily the node on which you ran the `clresource` command.

See Also The `clresource`(1CL) man page.

Bringing Online Resource Groups

To enable resources to begin providing HA services, you must perform the following operations:

- Enabling the resources in their resource groups
- Enabling the resource monitors
- Making the resource groups managed
- Bringing online the resource groups

You can perform these tasks individually or by using a single command.

After you bring online a resource group, it is configured and ready for use. If a resource or node fails, the RGM switches the resource group online on alternate nodes to maintain availability of the resource group.

▼ How to Bring Online Resource Groups

Perform this task from any cluster node.

1. **On a cluster member, become superuser or assume a role that provides `solaris.cluster.admin` RBAC authorization.**

2. **Type the command to bring online the resource groups.**

   - **If you have intentionally disabled a resource or a fault monitor that must remain disabled, type the following command:**

     ```
     # clresourcegroup online rg-list
     ```

     rg-list Specifies a comma-separated list of the names of the resource groups to bring online. The resource groups must exist. The list may contain one resource group name or more than one resource group name.

     You can omit the *rg-list* option. If you omit this option, all resource groups are brought online.

   - **If you require the resources and their fault monitors to be enabled when the resource groups are brought online, type the following command:**

     ```
     # clresourcegroup online -emM rg-list
     ```

     rg-list Specifies a comma-separated list of the names of the resource groups to bring online. The resource groups must exist. The list can contain one resource group name or more than one resource group name.

     You can omit the *rg-list* option. If you omit this option, all resource groups are brought online.

   Note – If any resource group that you are bringing online declares a strong affinity for other resource groups, this operation might fail. For more information, see "Distributing Online Resource Groups Among Cluster Nodes" on page 135.

3. **Verify that each resource group that you specified in Step 2 is online.**

   The output from this command indicates on which nodes each resource group is online.

   ```
   # clresourcegroup status
   ```

Example 2–11 Bringing Online a Resource Group

This example shows how to bring online the resource group `resource-group-1` and verify its status. All resources in this resource and their fault monitors are also enabled.

```
# clresourcegroup online -emM resource-group-1
# clresourcegroup status
```

Next Steps If you brought resource groups online *without* enabling their resources and fault monitors, enable the fault monitors of any resources that you require to be enabled. For more information, see "How to Enable a Resource Fault Monitor" on page 68.

See Also The `clresourcegroup(1CL)` man page.

Enabling a Resource

You can enable a resource that you neglected to enable when you brought online a resource group.

▼ How to Enable a Resource

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have created and have the name of the resource that you intend to enable.

1. **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.admin` **RBAC authorization.**

2. **Enable the resource.**

```
# clresource enable [-n node-zone-list] resource
```

`-n` *node-zone-list*

Specifies a comma-separated, ordered list of nodes on which to enable the resource. If you specify a global-cluster non-voting node, the format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*.

This list is optional. If you omit this list, the resource is enabled on all nodes in its resource group's node list.

Note – If you specify more than one node with the `-n` option, you can specify only one resource.

---

resource	Specifies the name of the resource that you want to enable.

3 **Verify that the resource has been enabled.**

```
# clresource status
```

The output from this command indicates the state of the resource that you have enabled.

See Also The `clresource(1CL)` man page.

Quiescing Resource Groups

To stop a resource group from continuously switching from one node to another when a `START` or `STOP` method fails, bring it to a quiescent state. To bring a resource group to a quiescent state, you issue the `clresourcegroup quiesce` command.

When you quiesce a resource group, resource methods that are executing are allowed to run until they are completed. If a serious problem occurs, you might need to quiesce a resource group immediately. To do so, you specify the `-k` command option, which kills the following methods:

- `Prenet_start`
- `Start`
- `Monitor_start`
- `Monitor_stop`
- `Stop`
- `Postnet_stop`

Note – The `Init`, `Fini` `Boot`, and `Update` methods are not killed when you specify this command option.

However, if you immediately quiesce a resource group by killing methods, you might leave one of its resources in an error state such as `Start_failed` or `Stop_failed`. You must clear these error states yourself.

▼ How to Quiesce a Resource Group

1 **Become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **Quiesce the resource group.**

```
# clresourcegroup quiesce resource-group
```

▼ How to Quiesce a Resource Group Immediately

1 **Become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Immediately quiesce the resource group.**

```
# clresourcegroup quiesce -k resource-group
```

The `Prenet_start`, `Start`, `Monitor_start`, `Monitor_stop`, `Stop`, and `Postnet_stop` methods that are associated with the resource group are killed immediately. The resource group is brought to a quiescent state.

The `clresourcegroup quiesce -k` command blocks until the specified resource group has reached a quiescent state.

Suspending and Resuming the Automatic Recovery Actions of Resource Groups

You can temporarily suspend the automatic recovery actions of a 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.

To suspend the automatic recovery actions of a resource group, you issue the `clresourcegroup suspend` command. To resume automatic recovery actions, you issue the `clresourcegroup resume` command.

When you suspend the automatic recovery actions of a resource group, you also bring the resource group to a quiescent state.

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. You can also still enable or disable individual resources in the resource group.

A dependency or affinity is suspended and not enforced when you suspend the automatic recovery actions of a resource group that does one of the following:

- Contains a resource that has a restart dependency on another resource

- Declares a strong positive or negative affinity for another resource group

When you suspend one of these categories of resource groups, Sun Cluster displays a warning that the dependency or affinity is suspended as well.

Note – Setting the `RG_system` property does not affect your ability to suspend or resume the automatic recovery actions of a resource group. However, if you suspend a resource group for which the `RG_system` property is set to `TRUE`, a warning message is produced. The `RG_system` property specifies that a resource group contains critical system services. If set to `TRUE`, the `RG_system` property prevents users from inadvertently stopping, deleting, or modifying a resource group or its resources.

Immediately Suspending Automatic Recovery by Killing Methods

When you suspend the automatic recovery actions of a resource group, resource methods that are executing are allowed to run until they are completed. If a serious problem occurs, you might need to suspend the automatic recovery actions of a resource group immediately. To do so, you specify the `-k` command option, which kills the following methods:

- `Prenet_start`
- `Start`
- `Monitor_start`
- `Monitor_stop`
- `Stop`
- `Postnet_stop`

Note – The `Init`, `Fini` `Boot`, and `Update` methods are not killed when you include this command option.

However, if you immediately suspend automatic recovery actions by killing methods, you might leave one of its resources in an error state such as `Start_failed` or `Stop_failed`. You must clear these error states yourself.

▼ How to Suspend the Automatic Recovery Actions of a Resource Group

1 **Become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Suspend the automatic recovery actions of the resource group.**

```
# clresourcegroup suspend resource-group
```

The resource group that you specify is not automatically started, restarted, or failed over until you resume automatic recovery actions. See "How to Resume the Automatic Recovery Actions of a Resource Group" on page 66.

▼ How to Suspend the Automatic Recovery Actions of a Resource Group Immediately

1 **Become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Immediately suspend the automatic recovery actions of the resource group.**

```
# clresourcegroup suspend -k resource-group
```

The `Prenet_start`, `Start`, `Monitor_start`, `Monitor_stop`, `Stop`, and `Postnet_stop` methods that are associated with the resource group are killed immediately. Automatic recovery actions of the resource group is suspended. The resource group is *not* automatically started, restarted, or failed over until you resume automatic recovery actions. See "How to Resume the Automatic Recovery Actions of a Resource Group" on page 66.

The `clresourcegroup suspend -k` command blocks until the specified resource group has reached a quiescent state.

▼ How to Resume the Automatic Recovery Actions of a Resource Group

1 **Become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Resume the automatic recovery actions of the resource group.**

```
# clresourcegroup resume resource-group
```

The resource group that you specify is automatically started, restarted, or failed over.

Disabling and Enabling Resource Monitors

The procedures in this section explain how to disable or enable resource fault monitors, not the resources themselves. A resource can continue to operate normally while its fault monitor is disabled. However, if the fault monitor is disabled and a data service fault occurs, automatic fault recovery is not initiated.

See the `clresource`(1CL) man page for additional information.

Note – Perform these procedures from any cluster node.

▼ How to Disable a Resource Fault Monitor

1. **On any cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2. **Disable the resource fault monitor.**

```
# clresource unmonitor [-n node-zone-list] resource
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes on which to unmonitor the resource. If you specify a global-cluster non-voting node, the format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the resource is unmonitored on all nodes in its resource group's node list. **Note –** If you specify more than one node with the `-n` option, you can specify only one resource.
resource	Specifies the name of the resource or resources.

3. **Run the** `clresource` **command on each cluster node and check for monitored fields (**`RS Monitored`**) to verify that the resource fault monitor has been disabled.**

```
# clresource show -v
```

Example 2–12 Disabling a Resource Fault Monitor

```
# clresource unmonitor resource-1
# clresource show -v
...
RS Monitored: no...
```

▼ How to Enable a Resource Fault Monitor

1 **On any cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **Enable the resource fault monitor.**

```
# clresource monitor [-n node-zone-list] resource
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes on which to monitor the resource. If you specify a global-cluster non-voting node, the format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global cluster, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the resource is monitored on all nodes in its resource group's node list. **Note –** If you specify more than one node with the `-n` option, you can specify only one resource.
resource	Specifies the name of the resource or resources.

3 **Run the** `clresource` **command on each cluster node and check for monitored fields** (`RS Monitored`) **to verify that the resource fault monitor has been enabled.**

```
# clresource show -v
```

Example 2–13 Enabling a Resource Fault Monitor

```
# clresource monitor resource-1
# clresource show -v
...
RS Monitored: yes...
```

Removing Resource Types

You do not need to remove resource types that are not in use. However, if you want to remove a resource type, follow this procedure.

Note – Perform this procedure from any cluster node.

▼ How to Remove a Resource Type

Removing a resource type involves disabling and removing all resources of that type in the cluster before unregistering the resource type.

Before You Begin To identify all instances of the resource type that you are removing, type the following command:

```
# clresourcetype show -v
```

1. **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2. **Disable each resource of the resource type that you are removing.**

   ```
   # clresource disable resource
   ```

   resource Specifies the name of the resource to disable.

3. **Remove each resource of the resource type that you are removing.**

   ```
   # clresource delete resource
   ```

   resource Specifies the name of the resource to remove.

4. **Unregister the resource type.**

   ```
   # clresourcetype unregister resource-type
   ```

   resource-type Specifies the name of the resource type to unregister.

5. **Verify that the resource type has been removed.**

   ```
   # clresourcetype show
   ```

Example 2–14 Removing a Resource Type

This example shows how to disable and remove all of the resources of a resource type (`resource-type-1`) and then unregister the resource type. In this example, `resource-1` is a resource of the resource type `resource-type-1`.

```
# clresource disable resource-1
# clresource delete resource-1
# clresourcetype unregister resource-type-1
```

See Also The following man pages:

- `clresource`(1CL)
- `clresourcetype`(1CL)

Removing Resource Groups

To remove a resource group, you must first remove all of the resources from the resource group.

Note – Perform this procedure from any cluster node.

▼ How to Remove a Resource Group

Before You Begin To identify all resources in the resource group that you are removing, type the following command:

```
# clresource show -v
```

1 **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Run the following command to switch the resource group offline.**

```
# clresourcegroup offline resource-group
```

resource-group Specifies the name of the resource group to take offline.

3 **Disable all of the resources in the resource group that you are removing.**

```
# clresource disable resource
```

resource Specifies the name of the resource to disable.

4 **Remove all of the resources from the resource group.**

For each resource, type the following command.

```
# clresource delete resource
```

resource	Specifies the name of the resource to be removed.

5 **Remove the resource group.**

```
# clresourcegroup delete resource-group
```

resource-group	Specifies the name of the resource group to be removed.

6 **Verify that the resource group has been removed.**

```
# clresourcegroup show
```

Example 2–15 Removing a Resource Group

This example shows how to remove a resource group (`resource-group-1`) after you have removed its resource (`resource-1`).

```
# clresourcegroup offline resource-group-1
# clresource disable resource-1
# clresource delete resource-1
# clresourcegroup delete resource-group-1
```

See Also The following man pages:

- `clresource`(1CL)
- `clresourcegroup`(1CL)

Removing Resources

Disable the resource before you remove it from a resource group.

Note – Perform this procedure from any cluster node.

▼ How to Remove a Resource

1 **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **Disable the resource that you are removing.**

```
# clrsource disable resource
```

resource	Specifies the name of the resource to disable.

3 **Remove the resource.**

```
# clresource delete resource
```

resource	Specifies the name of the resource to remove.

4 **Verify that the resource has been removed.**

```
# clresource show
```

Example 2–16 Removing a Resource

This example shows how to disable and remove a resource (`resource-1`).

```
# clresource disable resource-1
# clresource delete resource-1
```

See Also `clresource`(1CL)

Switching the Current Primary of a Resource Group

Use the following procedure to switch over a resource group from its current primary to another node that is to become the new primary.

▼ How to Switch the Current Primary of a Resource Group

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that the following conditions are met:

- You have the following information:
  - The name of the resource group that you are switching over
  - The names of the nodes where the resource group is to be brought online or to remain online

- The nodes where the resource group is to be brought online or to remain online are in the cluster.
- These nodes have been set up to be potential masters of the resource group that you are switching.

To see a list of potential primaries for the resource group, type the following command:

```
# clresourcegroup show -v
```

1 On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` **RBAC authorization.**

2 Switch the resource group to a new set of primaries.

```
# clresourcegroup switch [-n node-zone-list] resource-group
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of global-cluster non-voting nodes that can master this resource group. The resource group is switched offline on all of the other nodes. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the resource group is switched on all nodes in the resource group's node list.
resource-group	Specifies the name of the resource group to switch.

Note – If any resource group that you are switching declares a strong affinity for other resource groups, the attempt to switch might fail or be delegated. For more information, see "Distributing Online Resource Groups Among Cluster Nodes" on page 135.

---

3 Verify that the resource group has been switched to the new primary.

The output from this command indicates the state of the resource group that has been switched over.

```
# clresourcegroup status
```

Example 2–17 Switching a Resource Group to a New Primary

This example shows how to switch the resource group `resource-group-1` from its current primary `phys-schost-1` to the potential primary `phys-schost-2`.

1. To verify that the resource group is online on `phys-schost-1`, the following command is run:

```
phys-schost-1# clresourcegroup status

=== Cluster Resource Groups ===

    Group Name                    Node Name           Suspended        Status
    ----------                    ---------           ---------         ------

resource-group1                 phys-schost-1             No             Online
                                phys-schost-2             No             Offline
```

2. To perform the switch, the following command is run:

```
phys-schost-1# clresourcegroup switch -n phys-schost-2 resource-group-1
```

3. To verify that the group is switched to be online on `phys-schost-2`, the following command is run:

```
phys-schost-1# clresourcegroup status

=== Cluster Resource Groups ===

    Group Name                    Node Name           Suspended        Status
    ----------                    ---------           ---------         ------

resource-group1                 phys-schost-1             No             Offline
                                phys-schost-2             No             Online
```

See Also The `clresourcegroup(1CL)` page.

Disabling Resources and Moving Their Resource Group Into the UNMANAGED State

At times, you must bring a resource group into the UNMANAGED state before you perform an administrative procedure on it. Before you move a resource group into the UNMANAGED state, you must disable all of the resources that are part of the resource group and bring the resource group offline.

See the `clresourcegroup(1CL)` man page for additional information.

Note – Perform this procedure from any cluster node.

▼ How to Disable a Resource and Move Its Resource Group Into the UNMANAGED State

---

Note – When a shared address resource is disabled, the resource might still be able to respond to `ping(1M)` commands from some hosts. To ensure that a disabled shared address resource cannot respond to `ping` commands, you must bring the resource's resource group to the `UNMANAGED` state.

Before You Begin Ensure that you have the following information.

- The name of each resource to be disabled
- The name of the resource group to move into the `UNMANAGED` state

To determine the resource and resource group names that you need for this procedure, type:

```
# clresourcegroup show -v
```

1. **On any cluster member, become superuser or assume a role that provides `solaris.cluster.admin` RBAC authorization.**

2. **Disable all resources in the resource group.**

```
# clresource disable [-n node-zone-list] -g resource-group +
```

`-n` *node-zone-list* — Specifies a comma-separated, ordered list of nodes on which to disable the resource. If you specify a global-cluster non-voting node, the format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*.

This list is optional. If you omit this list, the resource is disabled on all nodes in its resource group's node list.

Note – If you specify more than one node with the `-n` option, you can specify only one resource.

---

3. **Switch the resource group offline.**

```
# clresourcegroup offline resource-group
```

resource-group — Specifies the name of the resource group to take offline.

4 **Move the resource group into the UNMANAGED state.**

```
# clresourcegroup unmanage resource-group
```

resource-group Specifies the name of the resource group to move into the UNMANAGED state.

5 **Verify that the resources are disabled and that the resource group is in the UNMANAGED state.**

```
# clrsourcegroup show resource-group
```

Example 2–18 Disabling a Resource and Moving Its Resource Group Into the UNMANAGED State

This example shows how to disable the resource (`resource-1`) and then move the resource group (`resource-group-1`) into the UNMANAGED state.

```
# clresource disable resource-1
# clresourcegroup offline resource-group-1
# clresourcegroup unmanage resource-group-1
# clresourcegroup show resource-group-1

=== Resource Groups and Resources ===

Resource Group:                                 resource-group-1
RG_description:                                 <NULL>
RG_mode:                                        Failover
RG_state:                                       Unmanaged
Failback:                                       False
Nodelist:                                       phys-schost-1 phys-schost-2

  --- Resources for Group resource-group-1 ---

  Resource:                                      resource-1
  Type:                                          SUNW.LogicalHostname:2
  Type_version:                                  2
  Group:                                         resource-group-1
  R_description:
  Resource_project_name:                         default
  Enabled{phys-schost-1}:                        False
  Enabled{phys-schost-2}:                        False
  Monitored{phys-schost-1}:                      True
  Monitored{phys-schost-2}:                      True
```

See Also The following man pages:

- `clresource`(1CL)
- `clresourcegroup`(1CL)

Displaying Resource Type, Resource Group, and Resource Configuration Information

Before you perform administrative procedures on resources, resource groups, or resource types, view the current configuration settings for these objects.

Note – You can view configuration settings for resources, resource groups, and resource types from any cluster node.

You can also use the `clresourcetype`, `clresourcegroup`, and `clresource` commands to check status information about specific resource types, resource groups, and resources. For example, the following command specifies that you want to view specific information about the resource `apache-1` only.

```
# clresource show apache-1
```

For more information, see the following man pages:

- `clresourcetype`(1CL)
- `clresourcegroup`(1CL)
- `clresource`(1CL)

Changing Resource Type, Resource Group, and Resource Properties

Sun Cluster defines standard properties for configuring resource types, resource groups, and resources. These standard properties are described in the following sections:

- "Resource Type Properties" on page 175
- "Resource Properties" on page 185
- "Resource Group Properties" on page 205

Resources also have extension properties, which are predefined for the data service that represents the resource. For a description of the extension properties of a data service, see the documentation for the data service.

To determine whether you can change a property, see the Tunable entry for the property in the description of the property.

The following procedures describe how to change properties for configuring resource types, resource groups, and resources.

▼ How to Change Resource Type Properties

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the following information.

- The name of the resource type to change.
- The name of the resource type property to change. For resource types, you can change only certain properties. To determine whether you can change a property, see the Tunable entry for the property in "Resource Type Properties" on page 175.

Note – You cannot change the `Installed_nodes` property explicitly. To change this property, specify the `-n` *installed-node-list* option of the `clresourcetype` command.

---

1. **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2. **Run the `clresourcetype` command to determine the name of the resource type that you need for this procedure.**

```
# clresourcetype show -v
```

3. **Change the resource type property.**

For resource types, you can change only certain properties. To determine whether you can change a property, see the Tunable entry for the property in "Resource Type Properties" on page 175.

```
# clresourcetype set -n installed-node-list \
[-p property=new-value]resource-type
```

`-n` *installed-node-list*	Specifies the names of nodes on which this resource type is installed.
`-p` *property=new-value*	Specifies the name of the standard property to change and the new value of the property. You cannot change the `Installed_nodes` property explicitly. To change this property, specify the `-n` *installed-node-list* option of the `clresourcetype` command.

4. **Verify that the resource type property has been changed.**

```
# clresourcetype show resource-type
```

Example 2–19 Changing a Resource Type Property

This example shows how to change the `SUNW.apache` property to define that this resource type is installed on the global-cluster voting nodes of (`phys-schost-1` and `phys-schost-2`).

```
# clresourcetype set -n phys-schost-1,phys-schost-2 SUNW.apache
# clresourcetype show SUNW.apache

Resource Type:                                    SUNW.apache:4
  RT_description:                                 Apache Web Server on Sun Cluster
  RT_version:                                     4
  API_version:                                    2
  RT_basedir:                                     /opt/SUNWscapc/bin
  Single_instance:                                False
  Proxy:                                          False
  Init_nodes:                                     All potential masters
  Installed_nodes:                                All
  Failover:                                       False
  Pkglist:                                        SUNWscapc
  RT_system:                                      False
```

▼ How to Change Resource Group Properties

This procedure explains how to change resource group properties. For a description of resource group properties, see "Resource Group Properties" on page 205.

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the following information.

- The name of the resource group to change
- The name of the resource group property to change and its new value

1 On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` **RBAC authorization.**

2 Change the resource group property.

```
# clresourcegroup set -p property=new-value resource-group
```

`-p` *property*	Specifies the name of the property to change
resource-group	Specifies the name of the resource group

3 **Verify that the resource group property has been changed.**

```
# clresourcegroup show resource-group
```

Example 2–20 Changing a Resource Group Property

This example shows how to change the `Failback` property for the resource group (`resource-group-1`).

```
# clresourcegroup set-p Failback=True resource-group-1
# clrsourcegroup show resource-group-1
```

▼ How to Change Resource Properties

This procedure explains how to change extension properties and standard properties of a resource.

- For a description of standard resource properties, see "Resource Properties" on page 185.
- For a description of the extension properties of a resource, see the documentation for the resource's resource type.

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the following information.

- The name of the resource with the property to change
- The name of the property to change

1 **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **View the current resource property settings.**

```
# clresource show -v resource
```

3 **Change the resource property.**

```
# clresource set -p standard-property=new-value | -p "extension-property \
[{node-specifier}]"=new-value  resource
```

`-p` *standard-property=new-value*
Specifies the name of the standard property to change.

`-p` "*extension-property*[{*node-specifier*}]"=*new-value*
Specifies the name of the extension property to change.

node-specifier is an *optional* qualifier to the -p and -x options. This qualifier indicates that the extension property or properties on *only* the specified node or nodes are to be set when the resource is created. The specified extension properties on other nodes in the cluster are not set. If you do not include *node-specifier*, the specified extension properties on all nodes in the cluster are set. You can specify a node name or a node identifier for *node-specifier*. Examples of the syntax of *node-specifier* include the following:

```
-p "myprop{phys-schost-1}"
```

The braces ({}) indicate that you are setting the specified extension property on only node phys-schost-1. For most shells, the double quotation marks (") are required.

You can also use the following syntax to set an extension property in two different global-cluster non-voting nodes on two different global-cluster voting nodes:

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

Note – The extension property that you specify with *node-specifier* must be declared in the RTR file as a per-node property. See Appendix B, "Standard Properties," for information about the Per_node resource property attribute.

---

resource
Specifies the name of the resource.

4 **Verify that the resource property has been changed.**

```
# clresource show -v resource
```

Example 2–21 Changing a Standard Resource Property

This example shows how to change the system-defined Start_timeout property for the resource (resource-1).

```
# clresource set -p start_timeout=30 resource-1
# clresource show -v resource-1
```

Example 2–22 Changing an Extension Resource Property

This example shows how to change an extension property (Log_level) for the resource (resource-1).

```
# clresource set -p Log_level=3 resource-1
# clresource show -v resource-1
```

▼ How to Modify a Logical Hostname Resource or a Shared Address Resource

By default, logical hostname resources and shared address resources use name services for name resolution. You might configure a cluster to use a name service that is running on the same cluster. During the failover of a logical hostname resource or a shared address resource, a name service that is running on the cluster might also be failing over. If the logical hostname resource or the shared address resource uses the name service that is failing over, the resource fails to fail over.

Note – Configuring a cluster to use a name server that is running on the same cluster might impair the availability of other services on the cluster.

To prevent such a failure to fail over, modify the logical hostname resource or the shared address resource to bypass name services. To modify the resource to bypass name services, set the `CheckNameService` extension property of the resource to `false`. You can modify the `CheckNameService` property at any time.

Note – If your version of the resource type is earlier than 2, you must upgrade the resource type before you attempt to modify the resource. For more information, see "Upgrading a Preregistered Resource Type" on page 89.

---

1. **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2. **Change the resource property.**

   ```
   # clresource set -p CheckNameService=false resource
   ```

`-p CheckNameService=false`	Sets the `CheckNameService` extension property of the resource to `false`.
resource	Specifies the name of the logical hostname resource or shared address resource that you are modifying.

Clearing the STOP_FAILED Error Flag on Resources

When the `Failover_mode` resource property is set to `NONE` or `SOFT`, a failure of the resource's `STOP` method causes the following effects:

- The individual resource goes into the `STOP_FAILED` state.
- The resource group that contains the resource goes into the `ERROR_STOP_FAILED` state.

In this situation, you cannot perform the following operations:

- Bringing online the resource group on any node
- Adding resources to the resource group
- Removing resources from the resource group
- Changing the properties of the resource group
- Changing the properties of resources in the resource group

▼ How to Clear the STOP_FAILED Error Flag on Resources

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the following information.

- The name of the node where the resource is STOP_FAILED
- The name of the resource and resource group that are in STOP_FAILED state

1 **On a cluster member, become superuser or assume a role that provides** solaris.cluster.modify **RBAC authorization.**

2 **Identify which resources have gone into the** STOP_FAILED **state and on which nodes.**

```
# clresource status
```

3 **Manually stop the resources and their monitors on the nodes on which they are in** STOP_FAILED **state.**

This step might require that you kill processes or run commands that are specific to resource types or other commands.

4 **Clear the** STOP_FAILED **error flag on the resources.**

```
# clresource clear -f STOP_FAILED -n nodelist resource
```

-f STOP_FAILED	Specifies the flag name.
-n *nodelist*	Specifies a comma-separated list of the names of the nodes where the resource is in the STOP_FAILED state. The list may contain one node name or more than one node name.
resource	Specifies the name of the resource.

5 **Check the resource group state on the nodes where you cleared the** STOP_FAILED **flag in Step 4.**

```
# clresourcegroup status
```

The resource group state should now be OFFLINE or ONLINE.

The resource group remains in the ERROR_STOP_FAILED state in the following combination of circumstances:

- The resource group was being switched offline when the STOP method failure occurred.
- The resource that failed to stop had a dependency on other resources in the resource group.

6 **If the resource group remains in the ERROR_STOP_FAILED state, correct the error as follows.**

   a. **Switch the resource group offline on the appropriate nodes.**

   ```
   # clresourcegroup offline resource-group
   ```

resource-group	Specifies the name of the resource group to switch offline.

   b. **Switch the resource group to the ONLINE state.**

See Also The following man pages:

- clresource(1CL)
- clresourcegroup(1CL)

Clearing the Start_failed Resource State

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 need only to execute a command that restarts the resource.

Use any one of the following procedures to clear this condition.

▼ How to Clear a Start_failed Resource State by Switching Over a Resource Group

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that the following conditions are met:

- You have the following information:
  - The name of the resource group that you are switching over
  - The name of the node on which to switch over the resource group
- The nodes where the resource group is to be brought online or to remain online are in the cluster .

1 **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **Switch the resource group to the new node.**

```
# clresourcegroup switch [-n node-zone-list] resource-group
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes that can master this resource group. This resource group is switched offline on all of the other nodes. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global cluster-voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the resource group is switched on all nodes in the resource group's node list.
resource-group	Specifies the name of the resource group to switch.

Note – If any resource group that you are switching declares a strong affinity for other resource groups, the attempt to switch might fail or be delegated. For more information, see "Distributing Online Resource Groups Among Cluster Nodes" on page 135.

3 **Verify that the resource group has been switched to the new node and that the** `Start_failed` **resource state is cleared.**

```
# clresourcegroup status
```

The output from this command indicates the state of the resource and the resource group that has been switched over.

Example 2–23 Clearing a `Start_failed` Resource State by Switching Over a Resource Group

This example shows how to clear a `Start_failed` resource state that has occurred on the `rscon` resource in the `resource-group-1` resource group. The command clears this condition by switching the resource group to the global cluster voting node `phys-schost-2`.

1. To verify that the resource is in the `Start_failed` resource state on `phys-schost-1`, the following command is run:

```
# clresource status

=== Cluster Resources ===

Resource Name              Node Name       Status        Message
--------------             ----------      -------        -------
 rscon                phys-schost-1      Faulted        Faulted
                      phys-schost-2      Offline         Offline

 hastor               phys-schost-1      Online         Online
                      phys-schost-2      Offline        Offline
```

2. To perform the switch, the following command is run:

   ```
   # clresourcegroup switch -n phys-schost-2 resource-group-1
   ```

3. To verify that the resource group is switched to be online on `phys-schost-2` and that the `Start_failed` resource status is cleared, the following command is run:

   ```
   # clresource status


   === Cluster Resources ===

   Resource Name              Node Name       Status        Message
   --------------             ----------      -------        -------
    rscon                phys-schost-1      Offline        Offline
                         phys-schost-2      Online         Online

    hastor               phys-schost-1      Online         Online
                         phys-schost-2      Offline        Offline
   ```

See Also The `clresourcegroup`(1CL) man page.

▼ How to Clear a `Start_failed` Resource State by Restarting a Resource Group

---

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that the following conditions are met:

- You have the following information:
  - The name of the resource group that you are restarting
  - The name of the node on which to restart the resource group

- The nodes where the resource group is to be brought online or to remain online are cluster nodes.

1 **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Restart the resource group.**

```
# clresourcegroup restart -n node resource-group
```

`-n` *node*	Specifies the name of the node on which the resource group is to be restarted. This resource group is switched offline on all of the other nodes.
resource-group	Specifies the name of the resource group to restart.

3 **Verify that the resource group has been restarted on the new node and that the `Start_failed` resource state is cleared.**

```
# clresourcegroup status
```

The output from this command indicates the state of the resource and the resource group that has been restarted.

Example 2–24 Clearing a `Start_failed` Resource State by Restarting a Resource Group

This example shows how to clear a `Start_failed` resource state that has occurred on the `rscon` resource in the `resource-group-1` resource group. The command clears this condition by restarting the resource group on the global-cluster voting node `phys-schost-1`.

1. To verify that the resource is in the `Start_failed` resource state on `phys-schost-1`, the following command is run:

```
# clresource status

=== Cluster Resources ===

Resource Name              Node Name       Status        Message
--------------        ----------      -------        -------
 rscon              phys-schost-1      Faulted        Faulted
                    phys-schost-2      Offline         Offline

 hastor             phys-schost-1      Online         Online
                    phys-schost-2      Offline        Offline
```

2. To restart the resource, the following command is run:

```
# clresourcegroup restart -n phys-schost-1 –g resource-group-1
```

3. To verify that the resource group is restarted on `phys-schost-1` and that the `Start_failed` resource status is cleared, the following command is run:

```
# clresource status

=== Cluster Resources ===

Resource Name              Node Name      Status        Message
--------------             ----------     -------        -------
 rscon                phys-schost-1      Offline        Offline
 rscon                phys-schost-2      Online         Online

 hastor               phys-schost-1      Online         Online
 hastor               phys-schost-2      Offline        Offline
```

See Also The clresourcegroup(1CL) man page.

▼ How to Clear a `Start_failed` Resource State by Disabling and Enabling a Resource

---

Note – Perform this procedure from any cluster node.

Before You Begin Ensure that you have the name of the resource that you are disabling and enabling.

1. **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2. **Disable and then enable the resource.**

```
# clresource disable resource
# clresource enable resource
```

resource	Specifies the name of the resource.

3. **Verify that the resource has been disabled and enabled and that the `Start_failed` resource state is cleared.**

```
# clresource status
```

The output from this command indicates the state of the resource that has been disabled and re-enabled.

Example 2–25 Clearing a `Start_failed` Resource State by Disabling and Enabling a Resource

This example shows how to clear a `Start_failed` resource state that has occurred on the `rscon` resource by disabling and enabling the resource.

1. To verify that the resource is in the `Start_failed` resource state, the following command is run:

   ```
   # clresource status

   === Cluster Resources ===

   Resource Name            Node Name      Status       Message
   --------------           ----------     -------       -------
    rscon              phys-schost-1      Faulted        Faulted
                       phys-schost-2      Offline         Offline

    hastor             phys-schost-1      Online         Online
                       phys-schost-2      Offline        Offline
   ```

2. To disable and re-enable the resource, the following commands are run:

   ```
   # clresource disable rscon
   # clresource enable rscon
   ```

3. To verify that the resource is re-enabled and that the `Start_failed` resource status is cleared, the following command is run:

   ```
   # clresource status


   === Cluster Resources ===

   Resource Name            Node Name      Status       Message
   --------------           ----------     -------       -------
    rscon              phys-schost-1      Online        Online
                       phys-schost-2      Offline       Offline

    hastor             phys-schost-1      Online         Online
                       phys-schost-2      Offline        Offline
   ```

See Also The `clresource`(1CL) man page.

Upgrading a Preregistered Resource Type

In Sun Cluster 3.1 9/04, the following preregistered resource types are enhanced:

- `SUNW.LogicalHostname`, which represents a logical hostname
- `SUNW.SharedAddress`, which represents a shared address

The purpose of these enhancements is to enable you to modify logical hostname resources and shared address resources to bypass name services for name resolution.

Upgrade these resource types if all conditions in the following list apply:

- You are upgrading from an earlier version of Sun Cluster.
- You need to use the new features of the resource types.

For general instructions that explain how to upgrade a resource type, see "Upgrading a Resource Type" on page 33. The information that you need to complete the upgrade of the preregistered resource types is provided in the subsections that follow.

Information for Registering the New Resource Type Version

The relationship between the version of each preregistered resource type and the release of Sun Cluster is shown in the following table. The release of Sun Cluster indicates the release in which the version of the resource type was introduced.

Resource Type	Resource Type Version	Sun ClusterRelease
`SUNW.LogicalHostname`	1.0	3.0
	2	3.1 9/04
`SUNW.SharedAddress`	1.0	3.0
	2	3.1 9/04

To determine the version of the resource type that is registered, use one command from the following list:

- `clresourcetype list`
- `clresourcetype list -v`

EXAMPLE 2–26 Registering a New Version of the `SUNW.LogicalHostname` Resource Type

This example shows the command for registering version 2 of the `SUNW.LogicalHostname` resource type during an upgrade.

```
# clresourcetype register SUNW.LogicalHostname:2
```

Information for Migrating Existing Instances of the Resource Type

The information that you need to migrate an instance of a preregistered resource type is as follows:

- You can perform the migration at any time.
- If you need to use the new features of the preregistered resource type, the required value of the `Type_version` property is `2`.
- If you are modifying the resource to bypass name services, set the `CheckNameService` extension property of the resource to `false`.

EXAMPLE 2–27 Migrating a Logical Hostname Resource

This example shows the command for migrating the logical hostname resource `lhostrs`. As a result of the migration, the resource is modified to bypass name services for name resolution.

```
# clresource set -p CheckNameService=false -p Type_version=2 lhostrs
```

Reregistering Preregistered Resource Types After Inadvertent Deletion

The resource types `SUNW.LogicalHostname` and `SUNW.SharedAddress` are preregistered. All of the logical hostname and shared address resources use these resource types. You never need to register these two resource types, but you might inadvertently delete them. If you have deleted resource types inadvertently, use the following procedure to reregister them.

Note – If you are upgrading a preregistered resource type, follow the instructions in "Upgrading a Preregistered Resource Type" on page 89 to register the new resource type version.

Note – Perform this procedure from any cluster node.

▼ How to Reregister Preregistered Resource Types After Inadvertent Deletion

● **Reregister the resource type.**

```
# clresourcetype register SUNW.resource-type
```

resource-type	Specifies the resource type to add (reregister). The resource type can be either `SUNW.LogicalHostname` or `SUNW.SharedAddress`.

Example 2–28 Reregistering a Preregistered Resource Type After Inadvertent Deletion

This example shows how to reregister the `SUNW.LogicalHostname` resource type.

```
# clresourcetype register SUNW.LogicalHostname
```

See Also The `clresourcetype`(1CL) man page.

Adding or Removing a Node to or From a Resource Group

The procedures in this section enable you to perform the following tasks.

- Configuring a cluster node to be an additional master of a resource group
- Removing a node from a resource group

The procedures are slightly different, depending on whether you plan to add or remove the node to or from a failover or scalable resource group.

Failover resource groups contain network resources that both failover and scalable services use. Each IP subnetwork connected to the cluster has its own network resource that is specified and included in a failover resource group. The network resource is either a logical hostname or a shared address resource. Each network resource includes a list of IPMP groups that it uses. For failover resource groups, you must update the complete list of IPMP groups for each network resource that the resource group includes (the `netiflist` resource property).

The procedure for scalable resource groups involves the following steps:

1. Repeating the procedure for failover groups that contain the network resources that the scalable resource uses
2. Changing the scalable group to be mastered on the new set of hosts

For more information, see the `clresourcegroup`(1CL) man page.

Note – Run either procedure from any cluster node.

Adding a Node to a Resource Group

The procedure to follow to add a node to a resource group depends on whether the resource group is a scalable resource group or a failover resource group. For detailed instructions, see the following sections:

- "How to Add a Node to a Scalable Resource Group" on page 93
- "How to Add a Node to a Failover Resource Group" on page 94

You must supply the following information to complete the procedure.

- The names and node IDs of all of the cluster nodes and names of zones
- The names of the resource groups to which you are adding the node
- The name of the IPMP group that is to host the network resources that are used by the resource group on all of the nodes

Also, be sure to verify that the new node is already a cluster member.

▼ How to Add a Node to a Scalable Resource Group

1 For each network resource that a scalable resource in the resource group uses, make the resource group where the network resource is located run on the new node.

See Step 1 through Step 5 in the following procedure for details.

2 Add the new node to the list of nodes that can master the scalable resource group (the `nodelist` resource group property).

This step overwrites the previous value of `nodelist`, and therefore you must include all of the nodes that can master the resource group here.

```
# clresourcegroup set [-n node-zone-list] resource-group
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes that can master this resource group. This resource group is switched offline on all of the other nodes. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the `Nodelist` property is set to all nodes in the cluster.
resource-group	Specifies the name of the resource group to which the node is being added.

3 **(Optional) Update the scalable resource's `Load_balancing_weights` property to assign a weight to the node that you are adding to the resource group.**

Otherwise, the weight defaults to `1`. See the `clresourcegroup`(1CL) man page for more information.

How to Add a Node to a Failover Resource Group

1 **Display the current node list and the current list of IPMP groups that are configured for each resource in the resource group.**

```
# clresourcegroup show -v resource-group | grep -i nodelist
# clresourcegroup show -v resource-group | grep -i netiflist
```

Note – The output of the command line for `nodelist` and `netiflist` identifies the nodes by node name. To identify node IDs, run the command `clnode show -v | grep -i` *node-id*.

---

2 **Update `netiflist` for the network resources that the node addition affects.**

This step overwrites the previous value of `netiflist`, and therefore you must include all the IPMP groups here.

```
# clresourcese set  -p netiflist=netiflist  network-resource
```

`-p netiflist=`*netiflist*	Specifies a comma-separated list that identifies the IPMP groups that are on each node. Each element in *netiflist* must be in the form of `netif@node`. `netif` can be given as an IPMP group name, such as `sc_ipmp0`. The node can be identified by the node name or node ID, such as `sc_ipmp0@1` or `sc_ipmp@phys-schost-1`.
network-resource	Specifies the name of the network resource (logical hostname or shared address) that is being hosted on the *netiflist* entries.

3 **If the `HAStoragePlus AffinityOn` extension property equals `True`, add the node to the appropriate disk set or device group.**

- **If you are using Solaris Volume Manager, use the `metaset` command.**

  ```
  # metaset -s disk-set-name -a -h node-name
  ```

`-s` *disk-set-name*	Specifies the name of the disk set on which the `metaset` command is to work
`-a`	Adds a drive or host to the specified disk set
`-h` *node-name*	Specifies the node to be added to the disk set

- **SPARC: If you are using Veritas Volume Manager, use the `clsetup` utility.**

  a. **On any active cluster member, start the `clsetup` utility.**

  ```
  # clsetup
  ```

  The Main Menu is displayed.

  b. **On the Main Menu, type the number that corresponds to the option for device groups and volumes.**

  c. **On the Device Groups menu, type the number that corresponds to the option for adding a node to a VxVM device group.**

  d. **Respond to the prompts to add the node to the VxVM device group.**

4 **Update the node list to include all of the nodes that can now master this resource group.**

This step overwrites the previous value of `nodelist`, and therefore you must include all of the nodes that can master the resource group here.

```
# clresourcegroup set [-n node-zone-list] resource-group
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of global-cluster non-voting nodes that can master this resource group. This resource group is switched offline on all the other nodes. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the `Nodelist` property is set to all nodes in the cluster.
resource-group	Specifies the name of the resource group to which the node is being added.

5 **Verify the updated information.**

```
# clresourcegroup show -vresource-group | grep -i nodelist
# clresourcegroup show -vresource-group | grep -i netiflist
```

Example 2–29 Adding a Node to a Resource Group

This example shows how to add a global-cluster voting node (`phys-schost-2`) to a resource group (`resource-group-1`) that contains a logical hostname resource (`schost-2`).

```
# clresourcegroup show -v resource-group-1 | grep -i nodelist
( Nodelist:    phys-schost-1 phys-schost-3
# clresourcegroup show -v resource-group-1 | grep -i netiflist
```

```
( Res property name: NetIfList
 Res property class: extension
 List of IPMP
interfaces on each node
 Res property type: stringarray
 Res property value: sc_ipmp0@1 sc_ipmp0@3

(Only nodes 1 and 3 have been assigned IPMP groups. You must add an IPMP group for node 2.)

# clresource set  -p netiflist=sc_ipmp0@1,sc_ipmp0@2,sc_ipmp0@3 schost-2
# metaset -s red -a -h phys-schost-2
# clresourcegroup set -n  phys-schost-1,phys-schost-2,phys-schost-3 resource-group-1
# clresourcegroup show -v resource-group-1 | grep -i nodelist
 Nodelist:     phys-schost-1 phys-schost-2
               phys-schost-3
# clresourcegroup show -v resource-group-1 | grep -i netiflist
 Res property value: sc_ipmp0@1 sc_ipmp0@2
                     sc_ipmp0@3
```

Removing a Node From a Resource Group

The procedure to follow to remove a node from a resource group depends on whether the resource group is a scalable resource group or a failover resource group. For detailed instructions, see the following sections:

- "How to Remove a Node From a Scalable Resource Group" on page 97
- "How to Remove a Node From a Failover Resource Group" on page 98
- "How to Remove a Node From a Failover Resource Group That Contains Shared Address Resources" on page 99

To complete the procedure, you must supply the following information.

- Node names and node IDs of all of the cluster nodes

  ```
  # clnode show -v | grep -i "Node ID"
  ```

- The name of the resource group or the names of the resource groups from which you plan to remove the node

  ```
  # clresourcegroup show | grep "Nodelist"
  ```

- Names of the IPMP groups that are to host the network resources that are used by the resource groups on all of the nodes

  ```
  # clresourcegroup show -v | grep "NetIfList.*value"
  ```

Additionally, be sure to verify that the resource group **is not mastered** on the node that you are removing. If the resource group **is mastered** on the node that you are removing, run the `clresourcegroup` command to switch the resource group offline from that node. The following `clresourcegroup` command brings the resource group offline from a given node, provided that *new-masters* does not contain that node.

```
# clresourcegroup switch -n new-masters resource-group
```

`-n` *new-masters*	Specifies the nodes that is now to master the resource group.
resource-group	Specifies the name of the resource group that you are switching . This resource group is mastered on the node that you are removing.

For more information, see the `clresourcegroup`(1CL) man page.

Caution – If you plan to remove a node from all the resource groups, and you use a scalable services configuration, first remove the node from the scalable resource groups. Then remove the node from the failover groups.

How to Remove a Node From a Scalable Resource Group

A scalable service is configured as two resource groups, as follows.

- One resource group is a scalable group that contains the scalable service resource.
- One resource group is a failover group that contains the shared address resources that the scalable service resource uses.

Additionally, the `RG_dependencies` property of the scalable resource group is set to configure the scalable group with a dependency on the failover resource group. For information about this property, see Appendix B, "Standard Properties."

For details about scalable service configuration, see *Sun Cluster Concepts Guide for Solaris OS*.

Removing a node from the scalable resource group causes the scalable service to no longer be brought online on that node. To remove a node from the scalable resource group, perform the following steps.

1 **Remove the node from the list of nodes that can master the scalable resource group (the `nodelist` resource group property).**

```
# clresourcegroup set [-n node-zone-list] scalable-resource-group
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes that can master this resource group. This resource group is switched offline on all the other nodes. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the

	global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the `Nodelist` property is set to all nodes in the cluster.
scalable-resource-group	Specifies the name of the resource group from which the node is being removed.

2 **(Optional) Remove the node from the failover resource group that contains the shared address resource.**

For details, see "How to Remove a Node From a Failover Resource Group That Contains Shared Address Resources" on page 99.

3 **(Optional) Update the `Load_balancing_weights` property of the scalable resource to remove the weight of the node that you are removing from the resource group.**

See Also The `clresourcegroup(1CL)` man page.

How to Remove a Node From a Failover Resource Group

Perform the following steps to remove a node from a failover resource group.

Caution – If you plan to remove a node from all of the resource groups, and you use a scalable services configuration, first remove the node from the scalable resource groups. Then use this procedure to remove the node from the failover groups.

Note – If the failover resource group contains shared address resources that scalable services use, see "How to Remove a Node From a Failover Resource Group That Contains Shared Address Resources" on page 99.

---

1 **Update the node list to include all of the nodes that can now master this resource group.**

This step removes the node and overwrites the previous value of the node list. Be sure to include all of the nodes that can master the resource group here.

```
# clresourcegroup set [-n node-zone-list] failover-resource-group
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes that can master this resource group. This resource group is switched offline on all the other nodes. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global

	cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the `Nodelist` property is set to all nodes in the cluster.
failover-resource-group	Specifies the name of the resource group from which the node is being removed.

2 **Display the current list of IPMP groups that are configured for each resource in the resource group.**

```
# clresourcegroup show -v failover-resource-group | grep -i netiflist
```

3 **Update `netiflist` for network resources that the removal of the node affects.**

This step overwrites the previous value of `netiflist`. Be sure to include all of the IPMP groups here.

```
# clresource set -p netiflist=netiflist  network-resource
```

Note – The output of the preceding command line identifies the nodes by node name. Run the command line `clnode show -v | grep -i "Node ID"` to find the node ID.

`-p netiflist=`*netiflist*	Specifies a comma-separated list that identifies the IPMP groups that are on each node. Each element in *netiflist* must be in the form of `netif@node`. `netif` can be given as an IPMP group name, such as `sc_ipmp0`. The node can be identified by the node name or node ID, such as `sc_ipmp0@1` or `sc_ipmp@phys-schost-1`.
network-resource	Specifies the name of the network resource that is hosted on the `netiflist` entries.

Note – Sun Cluster does not support the use of the adapter name for `netif`.

---

4 **Verify the updated information.**

```
# clresourcegroup show -vfailover-resource-group | grep -i nodelist
# clresourcegroup show -vfailover-resource-group | grep -i netiflist
```

▼ How to Remove a Node From a Failover Resource Group That Contains Shared Address Resources

In a failover resource group that contains shared address resources that scalable services use, a node can appear in the following locations.

- The node list of the failover resource group
- The `auxnodelist` of the shared address resource

To remove the node from the node list of the failover resource group, follow the procedure "How to Remove a Node From a Failover Resource Group" on page 98.

To modify the `auxnodelist` of the shared address resource, you must remove and recreate the shared address resource.

If you remove the node from the failover group's node list, you can continue to use the shared address resource on that node to provide scalable services. To continue to use the shared address resource, you must add the node to the `auxnodelist` of the shared address resource. To add the node to the `auxnodelist`, perform the following steps.

Note – You can also use the following procedure to **remove** the node from the `auxnodelist` of the shared address resource. To remove the node from the `auxnodelist`, you must delete and recreate the shared address resource.

---

1 **Switch the scalable service resource offline.**

2 **Remove the shared address resource from the failover resource group.**

3 **Create the shared address resource.**

Add the node ID or node name of the node that you removed from the failover resource group to the `auxnodelist`.

```
# clressharedaddress create -g failover-resource-group \
 -X new-auxnodelist shared-address
```

failover-resource-group	The name of the failover resource group that used to contain the shared address resource.
new-auxnodelist	The new, modified `auxnodelist` with the desired node added or removed.
shared-address	The name of the shared address.

Example – Removing a Node From a Resource Group

This example shows how to remove a node (`phys-schost-3`) from a resource group (`resource-group-1`) that contains a logical hostname resource (`schost-1`).

```
# clresourcegroup show -v resource-group-1 | grep -i nodelist
Nodelist:       phys-schost-1 phys-schost-2
                                             phys-schost-3
# clresourcegroup set -n phys-schost-1,phys-schost-2 resource-group-1
# clresourcegroup show -v resource-group-1 | grep -i netiflist
( Res property name: NetIfList
Res property class: extension
( List of IPMP
interfaces on each node
( Res property type: stringarray
 Res property value: sc_ipmp0@1 sc_ipmp0@2
                     sc_ipmp0@3

(sc_ipmp0@3 is the IPMP group to be removed.)

# clresource set  -p  netiflist=sc_ipmp0@1,sc_ipmp0@2 schost-1
# clresourcegroup show -v resource-group-1 | grep -i nodelist
Nodelist:       phys-schost-1 phys-schost-2
# clresourcegroup show -v resource-group-1 | grep -i netiflist
 Res property value: sc_ipmp0@1 sc_ipmp0@2
```

Migrating the Application From a Global-Cluster Voting Node to a Global-Cluster Non-Voting Node

You can migrate the application resources from a global-cluster voting node to a global-cluster non-voting node.

Note – The data services you want to migrate should be scalable and also be supported in global-cluster non-voting nodes

---

▼ How to Migrate the Application From a Global-Cluster Voting Node to a Global-Cluster Non-Voting Node

The procedure assumes a three node cluster with a global-cluster non-voting node created on each of the three nodes. The configuration directory that is made highly available using the `HAStoragePlus` resource should also be accessible from the global-cluster non-voting nodes.

1 **Create the failover resource group with the global-cluster voting node that holds the shared address that the scalable resource group is to use.**

```
# clresourcegroup create -n node1,node2,node3 sa-resource-group
```

sa-resource-group	Specifies your choice of the name of the failover resource group to add. This name must begin with an ASCII character.

2 Add the shared address resource to the failover resource group.

```
# clressharedaddress create -g sa-resource-group -h hostnamelist, … \
[-X auxnodelist] -N netiflist network-resource
```

`-g` *sa-resource-group*	Specifies the resource group name. In the node list of a shared address resource, do not specify more than one global-cluster non-voting node on the same global-cluster voting node. Specify the same list of *nodename:zonename* pairs as the node list of the scalable resource group.
`-h` *hostnamelist, …*	Specifies a comma-separated list of shared address hostnames.
`-X` *auxnodelist*	Specifies a comma-separated list of node names or IDs or zones that identify the cluster nodes that can host the shared address but never serve as primary if failover occurs. These nodes are mutually exclusive, with the nodes identified as potential masters in the resource group's node list. If no auxiliary node list is explicitly specified, the list defaults to the list of all cluster node names that are not included in the node list of the resource group that contains the shared address resource.

Note – To ensure that a scalable service runs in all global-cluster non-voting nodes that were created to master the service, the complete list of nodes must be included in the node list of the shared address resource group or the `auxnodelist` of the shared address resource. If all the global-cluster non-voting nodes are listed in the node list, the `auxnodelist` can be omitted.

`-N` *netiflist*	Specifies an optional, comma-separated list that identifies the IPMP groups that are on each node. Each element in *netiflist* must be in the form of `netif@node`. `netif` can be given as an IPMP group name, such as `sc_ipmp0`. The node can be identified by the node name or node ID, such as `sc_ipmp0@1` or `sc_ipmp@phys-schost-1`.

Note – Sun Cluster does not support the use of the adapter name for `netif`.

network-resource	Specifies an optional resource name of your choice.

3 Create the scalable resource group.

```
# clresourcegroup create\-p Maximum_primaries=m\-p Desired_primaries=n\
-n node1,node2,node3\
-p RG_dependencies=sa-resource-group resource-group-1
```

`-p Maximum_primaries=`*m*	Specifies the maximum number of active primaries for this resource group.
`-p Desired_primaries=`*n*	Specifies the number of active primaries on which the resource group should attempt to start.
resource-group-1	Specifies your choice of the name of the scalable resource group to add. This name must begin with an ASCII character.

4 **Create the `HAStoragePlus` resource `hastorageplus-1`, and define the filesystem mount points.**

```
# clresource create -g resource-group-1 -t SUNW.HAStoragePlus \
-p FilesystemMountPoints=/global/resource-group-1 hastorageplus-1
```

The resource is created in the enabled state.

5 **Register the resource type for the application.**

```
# clresourcetype register resource-type
```

resource-type	Specifies name of the resource type to add. See the release notes for your release of Sun Cluster to determine the predefined name to supply.

6 **Add the application resource to `resource-group-1`, and set the dependency to `hastorageplus-1`.**

```
# clresource create -g resource-group-1 -t SUNW.application \
[-p "extension-property[{node-specifier}]"=value, ...] -p Scalable=True \
-p Resource_dependencies=network-resource -p Port_list=port-number/protocol \
-p Resource_dependencies=hastorageplus-1 resource
```

7 **Bring the failover resource group online.**

```
# clresourcegroup online sa-resource-group
```

8 **Bring the scalable resource group online on all the nodes.**

```
# clresourcegroup online resource-group-1
```

9 **Install and boot** *zone1* **on each of the nodes,** *node1, node2, node3*.

10 **Bring the application resource group offline on two nodes (***node1, node2***).**

Note – Ensure the shared address is online on *node3*.

```
# clresourcegroup switch -n node3 resource-group-1
```

resource-group-1	Specifies the name of the resource group to switch.

11 **Update the `nodelist` property of the failover resource group to include the global-cluster non-voting node of the corresponding nodes removed from the node list.**

```
# clresourcegroup set -n node1:zone1,node2:zone1,node3 sa-resource-group
```

12 **Update the `nodelist` property of the application resource group to include the global-cluster non-voting node of the corresponding nodes removed from node list.**

```
# clresourcegroup set node1:zone1,node2:zone1,node3 resource-group-1
```

13 **Bring the failover resource group and application resource group online only on the newly added zones.**

Note – The failover resource group will be online only on *node1:zone1* and application resource group will be online only on *node1:zone1* and *node2:zone1*.

```
# clresourcegroup switch -n node1:zone1 sa-resource-group

# clresourcegroup switch -n node1:zone1,node2:zone1 resource-group-1
```

14 **Update the `nodelist` property of both the resource groups to include the global-cluster non-voting node of *node3* by removing the global node, *node3* from the list.**

```
# clresourcegroup set node1:zone1,node2:zone1,node3:zone1 sa-resource-group

# clresourcegroup set node1:zone1,node2:zone1,node3:zone1 resource-group-1
```

15 **Bring both the resource groups online on the global-cluster non-voting nodes.**

```
# clresourcegroup switch -n node1:zone1 sa-resource-group

# clresourcegroup switch -n node1:zone1,node2:zone1,node3:zone1 resource-group-1
```

Synchronizing the Startups Between Resource Groups and Device Groups

After a cluster boots or services fail over to another node, global devices and local and cluster file systems might require time to become available. However, a data service can run its `START` method before global devices and local and cluster file systems come online. If the data service depends on global devices or local and cluster file systems that are not yet online, the `START` method times out. In this situation, you must reset the state of the resource groups that the data service uses and restart the data service manually.

To avoid these additional administrative tasks, use the `HAStoragePlus` resource type. Add an instance of `HAStoragePlus` to all resource groups whose data service resources depend on global devices or local and cluster file systems. Instances of these resource types perform the following operations:

- Forcing the `START` method of the other resources in the same resource group to wait until global devices and local and cluster file systems become available.

To create an `HAStoragePlus` resource, see "How to Set Up the `HAStoragePlus` Resource Type for New Resources" on page 105.

Additional Administrative Tasks to Configure HAStoragePlus Resources for a Zone Cluster

When you configure HAStoragePlus resources for a zone cluster, you need to perform the following additional tasks before performing the steps for global cluster:

- While configuring file systems like UFS or standalone QFS in file system mount points, the file systems need to be configured to the zone cluster. For more information about configuring a file system to a zone cluster, see "How to Add a Local File System to a Zone Cluster" in *Sun Cluster Software Installation Guide for Solaris OS*.
- While configuring global devices in global device paths, the devices need to be configured to the zone cluster. For more information about configuring global devices to a zone cluster, see "Adding Storage Devices to a Zone Cluster" in *Sun Cluster Software Installation Guide for Solaris OS*.
- While configuring the ZFS file systems using Zpools, the ZFS pool needs to be configured to the zone cluster. For more information about configuring a ZFS file system to a zone cluster, see "How to Add a ZFS Storage Pool to a Zone Cluster" in *Sun Cluster Software Installation Guide for Solaris OS*.

Note – The cluster file system is not supported for Zone Cluster.

▼ How to Set Up the `HAStoragePlus` Resource Type for New Resources

In the following example, the resource group `resource-group-1` contains the following data services.

- Sun Java System Web Server, which depends on `/global/resource-group-1`
- Oracle, which depends on `/dev/global/dsk/d5s2`

- NFS, which depends on `dsk/d6`

Note – To create an `HAStoragePlus` resource with Solaris ZFS (Zettabyte File System) as a highly available local file system see"How to Set Up the `HAStoragePlus` Resource Type to Make a Local Solaris ZFS Highly Available" on page 117 section.

To create an `HAStoragePlus` resource `hastorageplus-1` for new resources in `resource-group-1`, read "Synchronizing the Startups Between Resource Groups and Device Groups" on page 104 and then perform the following steps.

To create an `HAStoragePlus` resource, see "Enabling Highly Available Local File Systems" on page 110.

1. **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` and `solaris.cluster.admin` RBAC authorizations.**

2. **Create the resource group `resource-group-1`.**

```
# clresourcegroup create resource-group-1
```

3. **Determine whether the resource type is registered.**

   The following command prints a list of registered resource types.

```
# clresourcetype show | egrep Type
```

4. **If you need to, register the resource type.**

```
# clresourcetype register SUNW.HAStoragePlus
```

5. **Create the `HAStoragePlus` resource `hastorageplus-1`, and define the filesystem mount points and global device paths.**

```
# clresource create -g resource-group-1 -t SUNW.HAStoragePlus \
-p GlobalDevicePaths=/dev/global/dsk/d5s2,dsk/d6 \
-p FilesystemMountPoints=/global/resource-group-1 hastorageplus-1
```

   `GlobalDevicePaths` can contain the following values.

   - Global device group names, such as `nfs-dg`, `dsk/d5`
   - Paths to global devices, such as `/dev/global/dsk/d1s2`, `/dev/md/nfsdg/dsk/d10`

   `FilesystemMountPoints` can contain the following values.

   - Mount points of local or cluster file systems, such as `/local-fs/nfs`, `/global/nfs`

Note – HAStoragePlus has a Zpools extension property that is used to configure ZFS file system storage pools and a ZpoolsSearchDir extension property that is used to specify the location to search for the devices of ZFS file system storage pools. The default value for the ZpoolsSearchDir extension property is /dev/dsk. The ZpoolsSearchDir extension property is similar to the -d option of the zpool(1M) command.

The resource is created in the enabled state.

6 **Add the resources (Sun Java System Web Server, Oracle, and NFS) to** resource-group-1**, and set their dependency to** hastorageplus-1**.**

For example, for Sun Java System Web Server, run the following command.

```
# clresource create  -g resource-group-1 -t SUNW.iws \
-p Confdir_list=/global/iws/schost-1 -p Scalable=False \
-p Network_resources_used=schost-1 -p Port_list=80/tcp \
-p Resource_dependencies=hastorageplus-1 resource
```

The resource is created in the enabled state.

7 **Verify that you have correctly configured the resource dependencies.**

```
# clresource show -v resource | egrep Resource_dependencies
```

8 **Set** resource-group-1 **to the** MANAGED **state, and bring** resource-group-1 **online.**

```
# clresourcegroup online -M resource-group-1
```

More Information

Affinity Switchovers

The HAStoragePlus resource type contains another extension property, AffinityOn, which is a Boolean that specifies whether HAStoragePlus must perform an affinity switchover for the global devices that are defined in GLobalDevicePaths and FileSystemMountPoints extension properties. For details, see the SUNW.HAStoragePlus(5) man page.

Note – The setting of the AffinityOn flag is ignored for scalable services. Affinity switchovers are not possible with scalable resource groups.

▼ How to Set Up the `HAStoragePlus` Resource Type for Existing Resources

Before You Begin Read "Synchronizing the Startups Between Resource Groups and Device Groups" on page 104.

1. **Determine whether the resource type is registered.**

   The following command prints a list of registered resource types.

   ```
   # clresourcetype show | egrep Type
   ```

2. **If you need to, register the resource type.**

   ```
   # clresourcetype register SUNW.HAStoragePlus
   ```

3. **Create the `HAStoragePlus` resource `hastorageplus-1`.**

   ```
   # clresource create -g resource-group \
   -t SUNW.HAStoragePlus -p GlobalDevicePaths= ... \
   -p FileSystemMountPoints=... -p AffinityOn=True hastorageplus-1
   ```

   The resource is created in the enabled state.

4. **Set up the dependency for each of the existing resources, as required.**

   ```
   # clresource set -p Resource_Dependencies=hastorageplus-1 resource
   ```

5. **Verify that you have correctly configured the resource dependencies.**

   ```
   # clresource show -v resource | egrep Resource_dependencies
   ```

Configuring an `HAStoragePlus` Resource for Cluster File Systems

When an `HAStoragePlus` resource is configured for cluster file systems and brought online, it ensures that these file systems are available. The cluster file system is supported on the UNIX File System (UFS) and Veritas File System (VxFS). The instructions in this section apply to HAStoragePlus resources with UFS. Use `HAStoragePlus` with local file systems if the data service is I/O intensive. See "How to Change the Cluster File System to Local File System in an `HAStoragePlus` Resource " on page 133 for information about how to change the file system of an `HAStoragePlus` resource.

Sample Entries in /etc/vfstab for Cluster File Systems

The following examples show entries in the /etc/vfstab file for global devices that are to be used for cluster file systems.

Note – The entries in the /etc/vfstab file for cluster file systems should contain the globalkeyword in the mount options.

EXAMPLE 2–30 Entries in /etc/vfstab for a Global Device With Solaris Volume Manager

This example shows entries in the /etc/vfstab file for a global device that uses Solaris Volume Manager.

```
/dev/md/kappa-1/dsk/d0   /dev/md/kappa-1/rdsk/d0
/global/local-fs/nfs ufs     5  yes     logging,global
```

EXAMPLE 2–31 Entries in /etc/vfstab for a Global Device With VxVM

This example shows entries in the /etc/vfstab file for a global device that uses VxVM.

```
/dev/vx/dsk/kappa-1/appvol    /dev/vx/rdsk/kappa-1/appvol
/global/local-fs/nfs vxfs     5 yes     log,global
```

▼ How to Set Up the HAStoragePlus Resource for Cluster File Systems

1. **On any node in the cluster, become superuser or assume a role that provides solaris.cluster.modify RBAC authorization.**

2. **Create a failover resource group.**

   ```
   # clresourcegroup create resource-group-1
   ```

3. **Register the HAStoragePlus resource type.**

   ```
   # clresourcetype register SUNW.HAStoragePlus
   ```

4. **Create the HAStoragePlus resource and define the filesystem mount points.**

   ```
   # clresource create -g resource-group -t SUNW.HAStoragePlus \
    -p FileSystemMountPoints="mount-point-list" hasp-resource
   ```

   The resource is created in the enabled state.

5 **Add the data service resources to** *resource-group-1***, and set their dependency to** *hasp-resource***.**

6 **Bring online and in a managed state the resource group that contains the** `HAStoragePlus` **resource.**

```
# clresourcegroup online -M resource-group-1
```

▼ How to Delete an `HAStoragePlus` Resource Type for Cluster File Systems

● **Disable and delete the** `HAStoragePlus` **resource configured for cluster file systems.**

```
# clresource delete -F -g resource-group -t SUNW.HAStoragePlus resource
```

Enabling Highly Available Local File Systems

Using a highly available local file system improves the performance of I/O intensive data services. To make a local file system highly available in a Sun Cluster environment, use the `HAStoragePlus` resource type.

You can specify global or local file systems. Global file systems are accessible from all nodes in a cluster. Local file systems are accessible from a single cluster node. Local file systems that are managed by a `SUNW.HAStoragePlus` resource are mounted on a single cluster node. 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 global-cluster non-voting node and the path in a global-cluster voting node , in this format:

Non-GlobalZonePath:*GlobalZonePath*

The global-cluster voting node path is optional. If you do not specify a global-cluster voting node path, Sun Cluster assumes that the path in the global-cluster non-voting node and in the global-cluster voting node are the same. If you specify the path as *Non-GlobalZonePath*:*GlobalZonePath*, you must specify *GlobalZonePath* in the global-cluster voting node'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 global-cluster non-voting node. To enable the `SUNW.HAStoragePlus` resource type to do this, you must create a mount point in the global-cluster voting node and in the global-cluster non-voting node. The `SUNW.HAStoragePlus` resource type makes the file system available to the global-cluster non-voting node by mounting the file system in the global cluster. The resource type then performs a loopback mount in the global-cluster non-voting node. Each file system mount point should have an equivalent entry in `/etc/vfstab` on all cluster nodes and in the global-cluster voting node. The `SUNW.HAStoragePlus` resource type does not check `/etc/vfstab` in global-cluster non-voting nodes.

Note – Local file systems include the UNIX File System (UFS), Quick File System (QFS), Veritas File System (VxFS), and Solaris ZFS (Zettabyte File System).

The instructions for each Sun Cluster data service that is I/O intensive explain how to configure the data service to operate with the `HAStoragePlus` resource type. For more information, see the individual Sun Cluster data service guides.

Note – Do *not* use the `HAStoragePlus` resource type to make a root file system highly available.

Sun Cluster provides the following tools for setting up the `HAStoragePlus` resource type to make local file systems highly available:

- **Sun Cluster Manager.** For more information, see the Sun Cluster Manager online help.
- **The `clsetup`(1CL) utility.**
- **Sun Cluster maintenance commands.**

Sun Cluster Manager and the `clsetup` utility enable you to add resources to the resource group interactively. Configuring these resources interactively reduces the possibility for configuration errors that might result from command syntax errors or omissions. Sun Cluster Manager and the `clsetup` utility ensure that all required resources are created and that all required dependencies between resources are set.

Configuration Requirements for Highly Available Local File Systems

Any file system on multihost disks must be accessible from any host that is directly connected to those multihost disks. To meet this requirement, configure the highly available local file system as follows:

- Ensure that the disk partitions of the local file system reside on global devices.
- Set the `AffinityOn` extension property of the `HAStoragePlus` resource that specifies these global devices to `True`.

The `Zpools` extension property of the `HAStoragePlus` resource ignores the `AffinityOn` extension property.

- Create the `HAStoragePlus` resource in a failover resource group.
- Ensure that the failback settings for the device groups and the resource group that contains the `HAStoragePlus` resource are identical.

Note – The use of a volume manager with the global devices for a highly available local file system is optional.

Format of Device Names for Devices Without a Volume Manager

If you are not using a volume manager, use the appropriate format for the name of the underlying storage device. The format to use depends on the type of storage device as follows:

- For block devices: `/dev/global/dsk/d`*D*`s`*S*
- For raw devices: `/dev/global/rdsk/d`*D*`s`*S*

The replaceable elements in these device names are as follows:

- *D* is an integer that specifies the device ID (DID) instance number.
- *S* is an integer that specifies the slice number.

Sample Entries in `/etc/vfstab` for Highly Available Local File Systems

The following examples show entries in the `/etc/vfstab` file for global devices that are to be used for highly available local file systems.

Note – Solaris ZFS (Zettabyte File System) does not use the `/etc/vfstab` file.

EXAMPLE 2–32 Entries in `/etc/vfstab` for a Global Device Without a Volume Manager

This example shows entries in the `/etc/vfstab` file for a global device on a physical disk without a volume manager.

```
/dev/global/dsk/d1s0       /dev/global/rdsk/d1s0
/global/local-fs/nfs  ufs     5  no     logging
```

EXAMPLE 2–33 Entries in `/etc/vfstab` for a Global Device With Solaris Volume Manager

This example shows entries in the `/etc/vfstab` file for a global device that uses Solaris Volume Manager.

```
/dev/md/kappa-1/dsk/d0   /dev/md/kappa-1/rdsk/d0
/global/local-fs/nfs ufs     5  no     logging
```

EXAMPLE 2–34 Entries in `/etc/vfstab` for a Global Device With VxVM

This example shows entries in the `/etc/vfstab` file for a global device that uses VxVM.

```
/dev/vx/dsk/kappa-1/appvol    /dev/vx/rdsk/kappa-1/appvol
/global/local-fs/nfs vxfs     5 no     log
```

▼ How to Set Up the `HAStoragePlus` Resource Type by Using the `clsetup` Utility

The following instructions explain how to how to set up the `HAStoragePlus` resource type by using the `clsetup` utility. Perform this procedure from any global-cluster voting node.

This procedure provides the long forms of the Sun Cluster maintenance commands. Most commands also have short forms. Except for the forms of the command names, the commands are identical. For a list of the commands and their short forms, see Appendix A, "Sun Cluster Object-Oriented Commands."

Before You Begin Ensure that the following prerequisites are met:

- Ensure that the required volumes, disk groups and file systems are created.

1. **Become superuser on any cluster voting node.**

2. **Start the `clsetup` utility.**

   ```
   # clsetup
   ```

   The `clsetup` main menu is displayed.

3. **Type the number that corresponds to the option for data services and press Return.**

   The Data Services menu is displayed.

4 **Type the number that corresponds to the option for configuring the file system and press Return.**

The `clsetup` utility displays the list of prerequisites for performing this task.

5 **Verify that the prerequisites are met, and press Return to continue.**

The `clsetup` utility displays a list of the cluster nodes that can master the highly available `HAStoragePlus` resource.

6 **Select the nodes that can master the highly available `HAStoragePlus` resource.**

- **To accept the default selection of all listed nodes in an arbitrary order, type `a` and press Return.**

- **To select a subset of the listed nodes, type a comma-separated or space-separated list of the numbers that correspond to the nodes. Then press Return.**

  Ensure that the nodes are listed in the order in which the nodes are to appear in the `HAStoragePlus` resource group's node list. The first node in the list is the primary node of this resource group.

- **To select all nodes in a particular order, type a comma-separated or space-separated ordered list of the numbers that correspond to the nodes and press Return.**

7 **To confirm your selection of nodes, type `d` and press Return.**

The `clsetup` utility displays a list of types of shared storage type where data is to be stored.

8 **Type the numbers that correspond to type of shared storage that you are using for storing the data and press Return.**

The `clsetup` utility displays the file system mount points that are configured in the cluster. If there are no existing mount points, the `clsetup` utility allows you to define a new mount point.

9 **Specify the default mount directory, the raw device path, the `Global Mount` option and the `Check File System Periodically` option and press Return.**

The `clsetup` utility returns you the properties of the mount point that the utility will create.

10 **To create the mount point, type `d` and press Return.**

The `clsetup` utility displays the available file system mount points.

Note – You can use the `c` option to define another new mount point.

---

11 **Select the file system mount points.**

- **To accept the default selection of all listed file system mount points in an arbitrary order, type** `a` **and press Return.**
- **To select a subset of the listed file system mount points, type a comma-separated or space-separated list of the numbers that correspond to the file system mount points and press Return.**

12 **To confirm your selection of nodes, type** `d` **and press Return.**

The `clsetup` utility displays the global disk sets and device groups that are configured in the cluster.

13 **Select the global device groups.**

- **To accept the default selection of all listed device groups in an arbitrary order, type** `a` **and press Return.**
- **To select a subset of the listed device groups, type a comma-separated or space-separated list of the numbers that correspond to the device groups and press Return.**

14 **To confirm your selection of nodes, type** `d` **and press Return.**

The `clsetup` utility displays the names of the Sun Cluster objects that the utility will create.

15 **If you require a different name for any Sun Cluster object, change the name as follows.**

a. **Type the number that corresponds to the name that you are changing and press Return.**

The `clsetup` utility displays a screen where you can specify the new name.

b. **At the New Value prompt, type the new name and press Return.**

The `clsetup` utility returns you to the list of the names of the Sun Cluster objects that the utility will create.

16 **To confirm your selection of Sun Cluster object names, type** `d` **and press Return.**

The `clsetup` utility displays information about the Sun Cluster configuration that the utility will create.

17 **To create the configuration, type** `c` **and Press Return.**

The `clsetup` utility displays a progress message to indicate that the utility is running commands to create the configuration. When configuration is complete, the `clsetup` utility displays the commands that the utility ran to create the configuration.

18 **(Optional) Type `q` and press Return repeatedly until you quit the `clsetup` utility.**

If you prefer, you can leave the `clsetup` utility running while you perform other required tasks before using the utility again. If you choose to quit `clsetup`, the utility recognizes your existing resource group when you restart the utility.

19 **Verify the `HAStoragePlus` resource has been created.**

Use the `clresource`(1CL) utility for this purpose. By default, the `clsetup` utility assigns the name *node_name*`-rg` to the resource group.

```
# clresource show node_name-rg
```

▼ How to Set Up the `HAStoragePlus` Resource Type to Make File Systems Highly Available Other Than Solaris ZFS

The following procedure explains how to set up the `HAStoragePlus` resource type to make file systems other than Solaris ZFS highly available.

1 **On any node in the global cluster, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Create a failover resource group.**

```
# clresourcegroup create resource-group
```

3 **Register the `HAStoragePlus` resource type.**

```
# clresourcetype register SUNW.HAStoragePlus
```

4 **Create the `HAStoragePlus` resource and define the file system mount points.**

```
# clresource create -g resource-group \
-t SUNW.HAStoragePlus -p FileSystemMountPoints=mount-point-list hasp-resource
```

5 **Bring online and in a managed state the resource group that contains the `HAStoragePlus` resource.**

```
# clresourcegroup online -M resource-group
```

Example 2–35 Setting Up the `HAStoragePlus` Resource Type to Make a UFS File System Highly Available for the Global Cluster

This example assumes that the file system */web-1* is configured to the `HAStoragePlus` resource to make the file system highly available for the global cluster.

```
phys-schost-1# vi /etc/vfstab
#device          device          mount           FS      fsck    mount   mount
#to mount        to fsck         point           type    pass    at boot options
#
# /dev/md/apachedg/dsk/d0 /dev/md/apachedg/rdsk/d0 /web-1 ufs 2 no logging
# clresourcegroup create hasp-rg
# clresourcetype register SUNW.HAStoragePlus
# clresource create -g hasp-rg -t SUNW.HAStoragePlus -p FileSystemMountPoints=/global/ufs-1 hasp-rs
# clresourcegroup online -M hasp-rg
```

Example 2–36 Setting Up the `HAStoragePlus` Resource Type to Make a UFS File System Highly Available for a Zone Cluster

This example assumes that the file system */web-1* is configured to the `HAStoragePlus` resource to make the file system highly available for a zone cluster *sczone*.

```
phys-schost-1# vi /etc/vfstab
#device          device          mount           FS      fsck    mount   mount
#to mount        to fsck         point           type    pass    at boot options
#
/dev/md/apachedg/dsk/d0 /dev/md/apachedg/rdsk/d0 /web-1 ufs 2 no logging
# clzonecluster configure sczone
clzc:sczone> add fs
clzc:sczone:fs> set dir=/web-1
clzc:sczone:fs> set special=/dev/md/apachedg/dsk/d0
clzc:sczone:fs> set raw=/dev/md/apachedg/rdsk/d0
clzc:sczone:fs> set type=ufs
clzc:sczone:fs> end
clzc:sczone:fs> exit
# clresourcegroup create -Z sczone hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# clresource create -Z sczone -g hasp-rg \
-t SUNW.HAStoragePlus -p FileSystemMountPoints=/global/ufs-1 hasp-rs
# clresourcegroup online -Z sczone -M hasp-rg
```

▼ How to Set Up the `HAStoragePlus` Resource Type to Make a Local Solaris ZFS Highly Available

You perform the following primary tasks to make a local Solaris ZFS (Zettabyte File System) highly available:

- Create a ZFS storage pool.
- Create a ZFS file system in that ZFS storage pool.
- Set up the `HAStoragePlus` resource that manages the ZFS storage pool.

This section describes how to complete both tasks.

1 **Create a ZFS storage pool.**

Caution – Do not add a configured quorum device to a ZFS storage pool. When a configured quorum device is added to a storage pool, the disk is relabeled as an EFI disk, the quorum configuration information is lost, and the disk no longer provides a quorum vote to the cluster. Once a disk is in a storage pool, you can configure that disk as a quorum device. Or, you can unconfigure the disk, add it to the storage pool, then reconfigure the disk as a quorum device.

Observe the following requirements when you create a ZFS storage pool in a Sun Cluster configuration:

- Ensure that all of the devices from which you create a ZFS storage pool are accessible from all nodes in the cluster. These nodes must be configured in the node list of the resource group to which the `HAStoragePlus` resource belongs.
- Ensure that the Solaris device identifier that you specify to the `zpool(1M)` command, for example `/dev/dsk/c0t0d0`, is visible to the `cldevice list -v` command.

Note – The ZFS storage pool can be created using a full disk or a disk slice. It is preferred to create a ZFS storage pool using a full disk by specifying a Solaris logical device as ZFS file system performs better by enabling the disk write cache. ZFS file system labels the disk with EFI when a full disk is provided.

See “Creating a ZFS Storage Pool” in *Solaris ZFS Administration Guide* for information about how to create a ZFS storage pool.

2 **In the ZFS storage pool that you just created, create a ZFS file system.**

You can create more than one ZFS file system in the same ZFS storage pool.

Note – `HAStoragePlus` does not support file systems created on ZFS file system volumes.

Do not place a ZFS file system in the `FilesystemMountPoints` extension property.

See “Creating a ZFS File System Hierarchy” in *Solaris ZFS Administration Guide* for information about how to create a ZFS file system in a ZFS storage pool.

3 **On any node in the cluster, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

4 **Create a failover resource group.**

```
# clresourcegroup create resource-group
```

5 **Register the `HAStoragePlus` resource type.**

```
# clresourcetype register SUNW.HAStoragePlus
```

6 **Create an `HAStoragePlus` resource for the local ZFS file system.**

```
# clresource create -g resource-group -t SUNW.HAStoragePlus \
-p Zpools=zpool -p ZpoolsSearchDir=/dev/did/dsk \
resource
```

The default location to search for devices of ZFS storage pools is `/dev/dsk`. It can be overridden by using the `ZpoolsSearchDir` extension property.

The resource is created in the enabled state.

7 **Bring online and in a managed state the resource group that contains the `HAStoragePlus` resource.**

```
# clresourcegroup online -M resource-group
```

Example 2–37 Setting Up the `HAStoragePlus` Resource Type to Make a Local ZFS Highly Available for the Global Cluster

The following example shows the commands to make a local ZFS file system highly available.

```
phys-schost-1% su
Password:
# cldevice list -v

DID Device          Full Device Path
----------          ----------------
d1                  phys-schost-1:/dev/rdsk/c0t0d0
d2                  phys-schost-1:/dev/rdsk/c0t1d0
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
    you can create a ZFS storage pool using a disk slice by specifying a Solaris device identifier:
# zpool create HAzpool c1t8d0s2
    or you can create a ZFS storage pool using disk slice by specifying a logical device identifier
# zpool create HAzpool /dev/did/dsk/d3s2
# zfs create HAzpool/export
# zfs create HAzpool/export/home
# clresourcegroup create hasp-rg
# clresourcetype register SUNW.HAStoragePlus
# clresource create -g hasp-rg -t SUNW.HAStoragePlus \
                    -p Zpools=HAzpool hasp-rs
# clresourcegroup online -M hasp-rg
```

Example 2–38 Setting Up the `HAStoragePlus` Resource Type to Make a Local ZFS Highly Available for a Zone Cluster

The following example shows the steps to make a local ZFS file system highly available in a zone cluster *sczone*.

```
phys-schost-1# cldevice list -v
# zpool create HAzpool c1t8d0
# zfs create HAzpool/export
# zfs create HAzpool/export/home
# clzonecluster configure sczone
clzc:sczone> add dataset
clzc:sczone:fs> set name=HAzpool
clzc:sczone:fs> end
clzc:sczone:fs> exit
# clresourcegroup create -Z sczone hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# clresource create -Z sczone -g hasp-rg \
-t SUNW.HAStoragePlus -p Zpools=HAzpool hasp-rs
# clresourcegroup online -Z -sczone -M hasp-rg
```

▼ How to Delete an `HAStoragePlus` Resource That Makes a Local Solaris ZFS Highly Available

● **Disable and delete the `HAStoragePlus` resource that makes a local Solaris ZFS (Zettabyte File System) highly available.**

```
# clresource delete -F -g resource-group -t SUNW.HAStoragePlus resource
```

Upgrading From `HAStorage` to `HAStoragePlus`

`HAStorage` is not supported in the current release of Sun Cluster software. Equivalent functionality is supported by `HAStoragePlus`. For instructions for upgrading from `HAStorage` to `HAStorage`, see the subsections that follow.

Note – The resource groups that have the HAStorage resource configured will be in STOP_FAILED state since HAStorage is no longer supported. Clear the ERROR_STOP_FAILED flag for the resource and follow the instructions to upgrade HAStorage to HAStoragePlus.

▼ How to Upgrade From HAStorage to HAStoragePlus When Using Device Groups or CFS

The following example uses a simple HA-NFS resource that is active with HAStorage. The ServicePaths are the disk group nfsdg and the AffinityOn property is True. Furthermore, the HA-NFS resource has Resource_Dependencies set to the HAStorage resource.

1 **Bring offline the resource group** nfs1-rg**.**

```
# clresourcegroup offline nfs1-rg
```

2 **Remove the dependencies the application resources has on** HAStorage**.**

```
# clresource set -p Resource_Dependencies="" nfsserver-rs
```

3 **Disable the** HAStorage **resource.**

```
# clresource disable nfs1storage-rs
```

4 **Remove the** HAStorage **resource from the application resource group.**

```
# clresource delete nfs1storage-rs
```

5 **Unregister the** HAStorage **resource type.**

```
# clresourcetype unregister SUNW.HAStorage
```

6 **Register the** HAStoragePlus **resource type.**

```
# clresourcetype register SUNW.HAStoragePlus
```

7 **Create the** HAStoragePlus **resource.**

Note – Instead of using the ServicePaths property of HAStorage, you must use the FilesystemMountPoints property or GlobalDevicePaths property of HAStoragePlus.

- **To specify the mount point of a file system, type the following command.**

  The FilesystemMountPoints extension property must match the sequence that is specified in /etc/vfstab.

  ```
  # clresource create -g nfs1-rg -t \
  SUNW.HAStoragePlus -p FilesystemMountPoints=/global/nfsdata -p \
  AffinityOn=True nfs1-hastp-rs
  ```

- **To specify global device paths, type the following command.**

  ```
  # clresource create -g nfs1-rg -t \
  SUNW.HAStoragePlus -p GlobalDevicePaths=nfsdg -p AffinityOn=True nfs1-hastp-rs
  ```

The resource is created in the enabled state.

8 **Disable the application server resource.**

```
# clresource disable nfsserver-rs
```

9 **Bring online the nfs1-rg group on a cluster node.**

```
# clresourcegroup online nfs1-rg
```

10 **Set up the dependencies between the application server and HAStoragePlus.**

```
# clresource set -p Resource_dependencies=nfs1-hastp-rs nfsserver-rs
```

11 **Bring online the nfs1-rg group on a cluster node.**

```
# clresourcegroup online -eM nfs1-rg
```

▼ How to Upgrade From HAStorage With CFS to HAStoragePlus With Highly Available Local File System

The following example uses a simple HA-NFS resource that is active with HAStorage. The ServicePaths are the disk group nfsdg and the AffinityOn property is True. Furthermore, the HA-NFS resource has Resource_Dependencies set to HAStorage resource.

1 **Remove the dependencies the application resource has on HAStorage resource.**

```
# clresource set -p Resource_Dependencies="" nfsserver-rs
```

2 **Disable the HAStorage resource.**

```
# clresource disable nfs1storage-rs
```

3 **Remove the HAStorage resource from the application resource group.**

```
# clresource delete nfs1storage-rs
```

4 **Unregister the HAStorage resource type.**

```
# clresourcetype unregister SUNW.HAStorage
```

5 **Modify /etc/vfstab to remove the global flag and change "mount at boot" to "no".**

6 **Create the HAStoragePlus resource.**

Note – Instead of using the ServicePaths property of HAStorage, you must use the FilesystemMountPoints property or GlobalDevicePaths property of HAStoragePlus.

- **To specify the mount point of a file system, type the following command.**

  The FilesystemMountPoints extension property must match the sequence that is specified in /etc/vfstab.

  ```
  # clresource create -g nfs1-rg -t \
  SUNW.HAStoragePlus -p FilesystemMountPoints=/global/nfsdata -p \
  AffinityOn=True nfs1-hastp-rs
  ```

- **To specify global device paths, type the following command.**

  ```
  # clresource create -g nfs1-rg -t \
  SUNW.HAStoragePlus -p GlobalDevicePaths=nfsdg -p AffinityOn=True nfs1-hastp-rs
  ```

The resource is created in the enabled state.

7 **Disable the application server resource.**

```
# clresource disable nfsserver-rs
```

8 **Bring online the nfs1-rg group on a cluster node.**

```
# clresourcegroup online nfs1-rg
```

9 **Set up the dependencies between the application server and HAStoragePlus.**

```
# clresource set -p Resource_dependencies=nfs1-hastp-rs nfsserver-rs
```

10 **Bring online the nfs1-rg group on a cluster node.**

```
# clresourcegroup online -eM nfs1-rg
```

Modifying Online the Resource for a Highly Available File System

You might need a highly available file system to remain available while you are modifying the resource that represents the file system. For example, you might need the file system to remain available because storage is being provisioned dynamically. In this situation, modify the resource that represents the highly available file system while the resource is online.

In the Sun Cluster environment, a highly available file system is represented by an `HAStoragePlus` resource. Sun Cluster enables you to modify an online `HAStoragePlus` resource as follows:

- Adding file systems to the `HAStoragePlus` resource
- Removing file systems from the `HAStoragePlus` resource

Note – Sun Cluster software does not enable you to rename a file system while the file system is online.

Note – When you remove the file systems configured in the `HAStoragePlus` resources for a zone cluster, you also need to remove the file system configuration from the zone cluster. For information about removing a file system from a zone cluster, see“How to Remove a File System From a Zone Cluster” in *Sun Cluster System Administration Guide for Solaris OS*.

▼ How to Add File Systems Other Than Solaris ZFS to an Online `HAStoragePlus` Resource

When you add a local or cluster file system to an `HAStoragePlus` resource, the `HAStoragePlus` resource automatically mounts the file system.

1. **On one node of the cluster, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2. **In the `/etc/vfstab` file on each node of the cluster, add an entry for the mount point of each file system that you are adding.**

   For each entry, set the mount at boot field and the mount options field as follows:

   - For local file systems
     - Set the mount at boot field to `no`.
     - Remove the `global` flag.

- For cluster file systems
  - If the file system is a cluster file system, set the mount options field to contain the `global` option.

3 **Retrieve the list of mount points for the file systems that the `HAStoragePlus` resource already manages.**

```
# scha_resource_get -O extension -R hasp-resource -G hasp-rg \
FileSystemMountPoints
```

`-R` *hasp-resource*	Specifies the `HAStoragePlus` resource to which you are adding file systems
`-G` *hasp-rg*	Specifies the resource group that contains the `HAStoragePlus` resource

4 **Modify the `FileSystemMountPoints` extension property of the `HAStoragePlus` resource to contain the following mount points:**

- The mount points of the file systems that the `HAStoragePlus` resource already manages
- The mount points of the file systems that you are adding to the `HAStoragePlus` resource

```
# clresource set -p FileSystemMountPoints="mount-point-list"  hasp-resource
```

`-p FileSystemMountPoints="`*mount-point-list*`"`
: Specifies a comma-separated list of mount points of the file systems that the `HAStoragePlus` resource already manages and the mount points of the file systems that you are adding. The format of each entry in the list is LocalZonePath:GlobalZonePathvooz. In this format, the global path is optional. If the global path is not specified, the global path is the same as the local path.

hasp-resource
: Specifies the `HAStoragePlus` resource to which you are adding file systems.

5 **Confirm that you have a match between the mount point list of the `HAStoragePlus` resource and the list that you specified in Step 4.**

```
# scha_resource_get -O extension -R hasp-resource -G hasp-rg \
 FileSystemMountPoints
```

`-R` *hasp-resource*	Specifies the `HAStoragePlus` resource to which you are adding file systems.
`-G` *hasp-rg*	Specifies the resource group that contains the `HAStoragePlus` resource.

6 **Confirm that the `HAStoragePlus` resource is online and not faulted.**

If the `HAStoragePlus` resource is online and faulted, validation of the resource succeeded, but an attempt by `HAStoragePlus` to mount a file system failed.

```
# clresource status hasp-resource
```

Example 2–39 Adding a File System to an Online `HAStoragePlus` Resource

This example shows how to add a file system to an online `HAStoragePlus` resource.

- The `HAStoragePlus` resource is named `rshasp` and is contained in the resource group `rghasp`.
- The `HAStoragePlus` resource named `rshasp` already manages the file system whose mount point is `/global/global-fs/fs`.
- The mount point of the file system that is to be added is `/global/local-fs/fs`.

The example assumes that the `/etc/vfstab` file on each cluster node already contains an entry for the file system that is to be added.

```
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints
STRINGARRAY
/global/global-fs/fs
# clresource set  \
-p FileSystemMountPoints="/global/global-fs/fs,/global/local-fs/fs"
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints rshasp
STRINGARRAY
/global/global-fs/fs
/global/local-fs/fs
# clresource status rshasp


=== Cluster Resources ===

Resource Name          Node Name      Status        Message
--------------        ----------      -------       --------
   rshasp               node46       Offline         Offline
                        node47       Online          Online
```

▼ How to Remove File Systems Other Than Solaris ZFS From an Online `HAStoragePlus` Resource

When you remove a file system from an `HAStoragePlus` resource, the `HAStoragePlus` resource treats a local file system differently from a cluster file system.

- The `HAStoragePlus` resource automatically unmounts a local file system.
- The `HAStoragePlus` resource does not unmount thecluster file system.

Caution – Before removing a file system from an online `HAStoragePlus` resource, ensure that no applications are using the file system. When you remove a file system from an online `HAStoragePlus` resource, the file system might be forcibly unmounted. If a file system that an application is using is forcibly unmounted, the application might fail or hang.

---

1 On one node of the cluster, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.

2 Retrieve the list of mount points for the file systems that the `HAStoragePlus` resource already manages.

```
# scha_resource_get -O extension -R hasp-resource -G hasp-rg \
FileSystemMountPoints
```

`-R` *hasp-resource*	Specifies the `HAStoragePlus` resource from which you are removing file systems.
`-G` *hasp-rg*	Specifies the resource group that contains the `HAStoragePlus` resource.

3 Modify the `FileSystemMountPoints` extension property of the `HAStoragePlus` resource to contain *only* the mount points of the file systems that are to remain in the `HAStoragePlus` resource.

```
# clresource set -p FileSystemMountPoints="mount-point-list"  hasp-resource
```

`-p FileSystemMountPoints="`*mount-point-list*`"`
Specifies a comma-separated list of mount points of the file systems that are to remain in the `HAStoragePlus` resource. This list must *not* include the mount points of the file systems that you are removing.

hasp-resource
Specifies the `HAStoragePlus` resource from which you are removing file systems.

4 Confirm that you have a match between the mount point list of the `HAStoragePlus` resource and the list that you specified in Step 3.

```
# scha_resource_get -O extension -R hasp-resource -G hasp-rg \
FileSystemMountPoints
```

`-R` *hasp-resource*	Specifies the `HAStoragePlus` resource from which you are removing file systems.
`-G` *hasp-rg*	Specifies the resource group that contains the `HAStoragePlus` resource.

5 **Confirm that the `HAStoragePlus` resource is online and not faulted.**

If the `HAStoragePlus` resource is online and faulted, validation of the resource succeeded, but an attempt by `HAStoragePlus` to unmount a file system failed.

```
# clresource status hasp-resource
```

6 **(Optional) From the `/etc/vfstab` file on each node of the cluster, remove the entry for the mount point of each file system that you are removing.**

Example 2–40 Removing a File System From an Online `HAStoragePlus` Resource

This example shows how to remove a file system from an online `HAStoragePlus` resource.

- The `HAStoragePlus` resource is named `rshasp` and is contained in the resource group `rghasp`.
- The `HAStoragePlus` resource named `rshasp` already manages the file systems whose mount points are as follows:
  - `/global/global-fs/fs`
  - `/global/local-fs/fs`
- The mount point of the file system that is to be removed is `/global/local-fs/fs`.

```
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints
STRINGARRAY
/global/global-fs/fs
/global/local-fs/fs
# clresource set -p FileSystemMountPoints="/global/global-fs/fs"
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints rshasp
STRINGARRAY
/global/global-fs/fs
 # clresource status rshasp


=== Cluster Resources ===

Resource Name          Node Name      Status        Message
--------------        ----------      -------       --------
   rshasp               node46       Offline         Offline
                        node47       Online          Online
```

▼ How to Add a Solaris ZFS Storage Pool to an Online `HAStoragePlus` Resource

When you add a Solaris ZFS (Zettabyte File System) storage pool to an online `HAStoragePlus` resource, the `HAStoragePlus` resource does the following:

- Imports the ZFS storage pool.
- Mounts all file systems in the ZFS storage pool.

1 On any node in the cluster, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.

2 Determine the ZFS storage pools that the `HAStoragePlus` resource already manages.

```
# clresource show -g hasp-resource-group -p Zpools hasp-resource
```

`-g` *hasp-resource-group*	Specifies the resource group that contains the `HAStoragePlus` resource.
hasp-resource	Specifies the `HAStoragePlus` resource to which you are adding the ZFS storage pool.

3 Add the new ZFS storage pool to the existing list of ZFS storage pools that the `HAStoragePlus` resource already manages.

```
# clresource set -p Zpools="zpools-list" hasp-resource
```

`-p` Zpools="*zpools-list*"	Specifies a comma-separated list of existing ZFS storage pool names that the `HAStoragePlus` resource already manages and the new ZFS storage pool name that you want to add.
hasp-resource	Specifies the `HAStoragePlus` resource to which you are adding the ZFS storage pool.

4 Compare the new list of ZFS storage pools that the `HAStoragePlus` resource manages with the list that you generated in Step 2.

```
# clresource show -g hasp-resource-group -p Zpools hasp-resource
```

`-g` *hasp-resource-group*	Specifies the resource group that contains the `HAStoragePlus` resource.
hasp-resource	Specifies the `HAStoragePlus` resource to which you added the ZFS storage pool.

5 **Confirm that the `HAStoragePlus` resource is online and not faulted.**

If the `HAStoragePlus` resource is online but faulted, validation of the resource succeeded. However, an attempt by the `HAStoragePlus` resource to import and mount the ZFS file system failed. In this case, you need to repeat the preceding set of steps.

```
# clresourcegroup status hasp-resource
```

▼ How to Remove a Solaris ZFS Storage Pool From an Online `HAStoragePlus` Resource

When you remove a Solaris ZFS (Zettabyte File System) storage pool from an online `HAStoragePlus` resource, the `HAStoragePlus` resource does the following:

- Unmounts the file systems in the ZFS storage pool.
- Exports the ZFS storage pool from the node.

1 **On any node in the cluster, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Determine the ZFS storage pools that the `HAStoragePlus` resource already manages.**

```
# clresource show -g hasp-resource-group -p Zpools hasp-resource
```

`-g` *hasp-resource-group*	Specifies the resource group that contains the `HAStoragePlus` resource.
hasp-resource	Specifies the `HAStoragePlus` resource from which you are removing the ZFS storage pool.

3 **Remove the ZFS storage pool from the list of ZFS storage pools that the `HAStoragePlus` resource currently manages.**

```
# clresource set -p Zpools="zpools-list" hasp-resource
```

`-p` Zpools="*zpools-list*"	Specifies a comma-separated list of ZFS storage pool names that the `HAStoragePlus` resource currently manages, minus the ZFS storage pool name that you want to remove.
hasp-resource	Specifies the `HAStoragePlus` resource from which you are removing the ZFS storage pool.

4 **Compare the new list of ZFS storage pools that the `HAStoragePlus` resource now manages with the list that you generated in Step 2.**

```
# clresource show -g hasp-resource-group -p Zpools hasp-resource
```

`-g` *hasp-resource-group*	Specifies the resource group that contains the `HAStoragePlus` resource.

hasp-resource	Specifies the `HAStoragePlus` resource from which you removed the ZFS storage pool.

5 **Confirm that the `HAStoragePlus` resource is online and not faulted.**

If the `HAStoragePlus` resource is online but faulted, validation of the resource succeeded. However, an attempt by the `HAStoragePlus` resource to unmount and export the ZFS file system failed. In this case, you need to repeat the preceding set of steps.

```
# clresourcegroup status SUNW.HAStoragePlus +
```

▼ How to Recover From a Fault After Modifying the `FileSystemMountPoints` Property of an `HAStoragePlus` Resource

If a fault occurs during a modification of the `FileSystemMountPoints` extension property, the status of the `HAStoragePlus` resource is online and faulted. After the fault is corrected, the status of the `HAStoragePlus` resource is online.

1 **Determine the fault that caused the attempted modification to fail.**

```
# clresource status hasp-resource
```

The status message of the faulty `HAStoragePlus` resource indicates the fault. Possible faults are as follows:

- The device on which the file system should reside does not exist.
- An attempt by the `fsck` command to repair a file system failed.
- The mount point of a file system that you attempted to add does not exist.
- A file system that you attempted to add cannot be mounted.
- A file system that you attempted to remove cannot be unmounted.

2 **Correct the fault that caused the attempted modification to fail.**

3 **Repeat the step to modify the `FileSystemMountPoints` extension property of the `HAStoragePlus` resource.**

```
# clresource set -p FileSystemMountPoints="mount-point-list" hasp-resource
```

`-p FileSystemMountPoints="`*mount-point-list*`"`
Specifies a comma-separated list of mount points that you specified in the unsuccessful attempt to modify the highly available file system

hasp-resource
Specifies the `HAStoragePlus` resource that you are modifying

4 **Confirm that the `HAStoragePlus` resource is online and not faulted.**

```
# clresource status
```

Example 2–41 Status of a Faulty `HAStoragePlus` Resource

This example shows the status of a faulty `HAStoragePlus` resource. This resource is faulty because an attempt by the `fsck` command to repair a file system failed.

```
# clresource status

  === Cluster Resources ===

  Resource Name     Node Name     Status       Status Message
  -------------     ----------    -------      -------------
  rshasp            node46        Offline      Offline
                    node47        Online       Online Faulted - Failed to fsck: /mnt.
```

▼ How to Recover From a Fault After Modifying the `Zpools` Property of an `HAStoragePlus` Resource

If a fault occurs during a modification of the `Zpools` extension property, the status of the `HAStoragePlus` resource is online and faulted. After the fault is corrected, the status of the `HAStoragePlus` resource is online.

1 **Determine the fault that caused the attempted modification to fail.**

```
# clresource status hasp-resource
```

The status message of the faulty `HAStoragePlus` resource indicates the fault. Possible faults are as follows:

- The ZFS pool *zpool* failed to import.
- The ZFS pool *zpool* failed to export.

2 **Correct the fault that caused the attempted modification to fail.**

3 **Repeat the step to modify the `Zpools` extension property of the `HAStoragePlus` resource.**

```
# clresource set -p Zpools="zpools-list" hasp-resource
```

`-p Zpools="`*zpools-list*`"`	Specifies a comma-separated list of ZFS storage pool names that the `HAStoragePlus` currently manages, minus the ZFS storage pool name that you want to remove.
hasp-resource	Specifies the `HAStoragePlus` resource that you are modifying

4 **Confirm that the `HAStoragePlus` resource is online and not faulted.**

```
# clresource status
```

Example 2–42 Status of a Faulty `HAStoragePlus` Resource

This example shows the status of a faulty `HAStoragePlus` resource. This resource is faulty because the ZFS pool *zpool* failed to import.

```
# clresource status hasp-resource

  === Cluster Resources ===

  Resource Name     Node Name     Status           Status Message
  --------------    ----------    -------          -------------
  hasp-resource     node46        Online           Faulted - Failed to import:hazpool
                    node47        Offline          Offline
```

Changing the Cluster File System to a Local File System in an `HAStoragePlus` Resource

You can change the file system of an `HAStoragePlus` resource from a cluster file system to a local file system.

▼ How to Change the Cluster File System to Local File System in an `HAStoragePlus` Resource

1 **Bring the failover resource group offline.**

```
# clresourcegroup offline resource-group
```

2 **Display the `HAStoragePlus` resource.**

```
# clresource show -g resource-group -t SUNW.HAStoragePlus
```

3 **Retrieve the list of mount points for each resource.**

```
# clresource show -p FilesystemMountPoints  hastorageplus-resource
```

4 **Unmount the cluster file system.**

```
# umount mount-points
```

5 Modify the `/etc/vfstab` entry of the mount points on all the nodes configured in the node list of the resource group.

- **Remove the `global` keyword from the mount options.**
- **Modify the `mount at boot` option from `yes` to `no`.**

Repeat the steps for all the cluster file systems of all the `HAStoragePlus` resources configured in the resource group.

6 Bring the resource group online.

```
# clresourcegroup online -M resource-group
```

Upgrading the `HAStoragePlus` Resource Type

In Sun Cluster 3.1 9/04, the `HAStoragePlus` resource type is enhanced to enable you to modify highly available file systems online. Upgrade the `HAStoragePlus` resource type if all conditions in the following list apply:

- You are upgrading from an earlier version of Sun Cluster.
- You need to use the new features of the `HAStoragePlus` resource type.

For general instructions that explain how to upgrade a resource type, see "Upgrading a Resource Type" on page 33. The information that you need to complete the upgrade of the `HAStoragePlus` resource type is provided in the subsections that follow.

Information for Registering the New Resource Type Version

The relationship between a resource type version and the release of Sun Cluster is shown in the following table. The release of Sun Cluster indicates the release in which the version of the resource type was introduced.

Resource Type Version	Sun ClusterRelease
1.0	3.0 5/02
2	3.1 9/04
4	3.2
6	3.2 2/08

To determine the version of the resource type that is registered, use one command from the following list:

- `clresourcetype list`
- `clresourcetype list -v`

The RTR file for this resource type is `/usr/cluster/lib/rgm/rtreg/SUNW.HAStoragePlus`.

Information for Migrating Existing Instances of the Resource Type

The information that you need to migrate instances of the `HAStoragePlus` resource type is as follows:

- You can perform the migration at any time.
- If you need to use the new features of the `HAStoragePlus` resource type, the required value of the `Type_version` property is `4`.

Distributing Online Resource Groups Among Cluster Nodes

For maximum availability or optimum performance, some combinations of services require a specific distribution of online resource groups among cluster nodes. Distributing online resource groups involves creating affinities between resource groups for the following purposes:

- Enforcing the required distribution when the resource groups are first brought online
- Preserving the required distribution after an attempt to fail over or switch over a resource group

This section provides the following examples of how to use resource group affinities to distribute online resource groups among cluster nodes:

- Enforcing collocation of a resource group with another resource group
- Specifying a preferred collocation of a resource group with another resource group
- Balancing the load of a set of resource groups
- Specifying that a critical service has precedence
- Delegating the failover or switchover of a resource group
- Combining affinities between resource groups to specify more complex behavior

Resource Group Affinities

An affinity between resource groups restricts on which nodes the resource groups may be brought online simultaneously. In each affinity, a source resource group declares an affinity for a target resource group or several target resource groups. To create an affinity between resource groups, set the `RG_affinities` resource group property of the source as follows:

`-p RG_affinities=`*affinity-list*

affinity-list
: Specifies a comma-separated list of affinities between the source resource group and a target resource group or several target resource groups. You may specify a single affinity or more than one affinity in the list.

  Specify each affinity in the list as follows:

  operator target-rg

  Note – Do not include a space between *operator* and *target-rg*.

  ---

  operator
  : Specifies the type of affinity that you are creating. For more information, see Table 2–2.

  target-rg
  : Specifies the resource group that is the target of the affinity that you are creating.

TABLE 2–2 Types of Affinities Between Resource Groups

Operator	Affinity Type	Effect
+	Weak positive	If possible, the source is brought online on a node or on nodes where the target is online or starting. However, the source and the target are allowed to be online on different nodes.
++	Strong positive	The source is brought online only on a node or on nodes where the target is online or starting. The source and the target are *not* allowed to be online on different nodes.
-	Weak negative	If possible, the source is brought online on a node or on nodes where the target is *not* online or starting. However, the source and the target are allowed to be online on the same node.
- -	Strong negative	The source is brought online only on a node or on nodes where the target is not online. The source and the target are *not* allowed to be online on the same node.
+++	Strong positive with failover delegation	Same as strong positive, except that an attempt by the source to fail over is delegated to the target. For more information, see "Delegating the Failover or Switchover of a Resource Group" on page 141.

Weak affinities take precedence over `Nodelist` preference ordering.

The current state of other resource groups might prevent a strong affinity from being satisfied on any node. In this situation, the resource group that is the source of the affinity remains offline. If other resource groups' states change to enable the strong affinities to be satisfied, the resource group that is the source of the affinity comes back online.

Note – Use caution when declaring a strong affinity on a source resource group for more than one target resource group. If all declared strong affinities cannot be satisfied, the source resource group remains offline.

Enforcing Collocation of a Resource Group With Another Resource Group

A service that is represented by one resource group might depend so strongly on a service in a second resource group that both services must run on the same node. For example, an application that is comprised of multiple interdependent service daemons might require that all daemons run on the same node.

In this situation, force the resource group of the dependent service to be collocated with the resource group of the other service. To enforce collocation of a resource group with another resource group, declare on the resource group a strong positive affinity for the other resource group.

```
# clresourcegroup set|create -p RG_affinities=++target-rg source-rg
```

source-rg
: Specifies the resource group that is the source of the strong positive affinity. This resource group is the resource group *on* which you are declaring a strong positive affinity for another resource group.

`-p RG_affinities=++`*target-rg*
: Specifies the resource group that is the target of the strong positive affinity. This resource group is the resource group *for* which you are declaring a strong positive affinity.

A resource group follows the resource group for which it has a strong positive affinity. If the target resource group is relocated to a different node, the source resource group automatically switches to the same node as the target. However, a resource group that declares a strong positive affinity is prevented from failing over to a node on which the target of the affinity is not already running.

Note – Only failovers that are initiated by a resource monitor are prevented. If a node on which the source resource group and target resource group are running fails, both resource groups fail over to the same surviving node.

For example, a resource group `rg1` declares a strong positive affinity for resource group `rg2`. If `rg2` fails over to another node, `rg1` also fails over to that node. This failover occurs even if all the resources in `rg1` are operational. However, if a resource in `rg1` attempts to fail over `rg1` to a node where `rg2` is not running, this attempt is blocked.

The source of a strong positive affinity might be offline on all nodes when you bring online the target of the strong positive affinity. In this situation, the source of the strong positive affinity is automatically brought online on the same node as the target.

For example, a resource group `rg1` declares a strong positive affinity for resource group `rg2`. Both resource groups are initially offline on all nodes. If an administrator brings online `rg2` on a node, `rg1` is automatically brought online on the same node.

You can use the `clresourcegroup suspend` command to prevent a resource group from being brought online automatically due to strong affinities or cluster reconfiguration.

If you require a resource group that declares a strong positive affinity to be allowed to fail over, you must delegate the failover. For more information, see "Delegating the Failover or Switchover of a Resource Group" on page 141.

EXAMPLE 2–43 Enforcing Collocation of a Resource Group With Another Resource Group

This example shows the command for modifying resource group `rg1` to declare a strong positive affinity for resource group `rg2`. As a result of this affinity relationship, `rg1` is brought online only on nodes where `rg2` is running. This example assumes that both resource groups exist.

```
# clresourcegroup set -p RG_affinities=++rg2 rg1
```

Specifying a Preferred Collocation of a Resource Group With Another Resource Group

A service that is represented by one resource group might use a service in a second resource group. As a result, these services run most efficiently if they run on the same node. For example, an application that uses a database runs most efficiently if the application and the database run on the same node. However, the services can run on different nodes because the reduction in efficiency is less disruptive than additional failovers of resource groups.

In this situation, specify that both resource groups should be collocated if possible. To specify preferred collocation of a resource group with another resource group, declare on the resource group a weak positive affinity for the other resource group.

```
# clresourcegroup set|create -p RG_affinities=+target-rg source-rg
```

source-rg
Specifies the resource group that is the source of the weak positive affinity. This resource group is the resource group *on* which you are declaring a weak positive affinity for another resource group.

`-p RG_affinities=+`*target-rg*
Specifies the resource group that is the target of the weak positive affinity. This resource group is the resource group *for* which you are declaring a weak positive affinity.

By declaring a weak positive affinity on one resource group for another resource group, you increase the probability of both resource groups running on the same node. The source of a weak positive affinity is first brought online on a node where the target of the weak positive affinity is already running. However, the source of a weak positive affinity does not fail over if a resource monitor causes the target of the affinity to fail over. Similarly, the source of a weak positive affinity does not fail over if the target of the affinity is switched over. In both situations, the source remains online on the node where the source is already running.

Note – If a node on which the source resource group and target resource group are running fails, both resource groups are restarted on the same surviving node.

EXAMPLE 2–44 Specifying a Preferred Collocation of a Resource Group With Another Resource Group

This example shows the command for modifying resource group `rg1` to declare a weak positive affinity for resource group `rg2`. As a result of this affinity relationship, `rg1` and `rg2` are first brought online on the same node. But if a resource in `rg2` causes `rg2` to fail over, `rg1` remains online on the node where the resource groups were first brought online. This example assumes that both resource groups exist.

```
# clresourcegroup set -p RG_affinities=+rg2 rg1
```

Distributing a Set of Resource Groups Evenly Among Cluster Nodes

Each resource group in a set of resource groups might impose the same load on the cluster. In this situation, by distributing the resource groups evenly among cluster nodes, you can balance the load on the cluster.

To distribute a set of resource groups evenly among cluster nodes, declare on each resource group a weak negative affinity for the other resource groups in the set.

```
# clresourcegroup set|create -p RG_affinities=neg-affinity-list  source-rg
```

source-rg
Specifies the resource group that is the source of the weak negative affinity. This resource group is the resource group *on* which you are declaring a weak negative affinity for other resource groups.

`-p RG_affinities=`*neg-affinity-list*
Specifies a comma-separated list of weak negative affinities between the source resource group and the resource groups that are the target of the weak negative affinity. The target resource groups are the resource groups *for* which you are declaring a weak negative affinity.

By declaring a weak negative affinity on one resource group for other resource groups, you ensure that a resource group is always brought online on the most lightly loaded node in the cluster. The fewest other resource groups are running on that node. Therefore, the smallest number of weak negative affinities are violated.

EXAMPLE 2–45 Distributing a Set of Resource Groups Evenly Among Cluster Nodes

This example shows the commands for modifying resource groups `rg1`, `rg2`, `rg3`, and `rg4` to ensure that these resource groups are evenly distributed among the available nodes in the cluster. This example assumes that resource groups `rg1`, `rg2`, `rg3`, and `rg4` exist.

```
# clresourcegroup set -p RG_affinities=-rg2,-rg3,-rg4 rg1
# clresourcegroup set -p RG_affinities=-rg1,-rg3,-rg4 rg2
# clresourcegroup set -p RG_affinities=-rg1,-rg2,-rg4 rg3
# clresourcegroup set -p RG_affinities=-rg1,-rg2,-rg3 rg4
```

Specifying That a Critical Service Has Precedence

A cluster might be configured to run a combination of mission-critical services and noncritical services. For example, a database that supports a critical customer service might run in the same cluster as noncritical research tasks.

To ensure that the noncritical services do not affect the performance of the critical service, specify that the critical service has precedence. By specifying that the critical service has precedence, you prevent noncritical services from running on the same node as the critical service.

When all nodes are operational, the critical service runs on a different node from the noncritical services. However, a failure of the critical service might cause the service to fail over to a node where the noncritical services are running. In this situation, the noncritical services are taken offline immediately to ensure that the computing resources of the node are fully dedicated to the mission-critical service.

To specify that a critical service has precedence, declare on the resource group of each noncritical service a strong negative affinity for the resource group that contains the critical service.

```
# clresourcegroup set|create -p RG_affinities=--critical-rg noncritical-rg
```

noncritical-rg
Specifies the resource group that contains a noncritical service. This resource group is the resource group *on* which you are declaring a strong negative affinity for another resource group.

`-p RG_affinities=--`*critical-rg*
Specifies the resource group that contains the critical service. This resource group is the resource group *for* which you are declaring a strong negative affinity.

A resource group moves away from a resource group for which it has a strong negative affinity.

The source of a strong negative affinity might be offline on all nodes when you take offline the target of the strong negative affinity. In this situation, the source of the strong negative affinity is automatically brought online. In general, the resource group is brought online on the most preferred node, based on the order of the nodes in the node list and the declared affinities.

For example, a resource group `rg1` declares a strong negative affinity for resource group `rg2`. Resource group `rg1` is initially offline on all nodes, while resource group `rg2` is online on a node. If an administrator takes offline `rg2`, `rg1` is automatically brought online.

You can use the `clresourcegroup suspend` command to prevent the source of a strong negative affinity from being brought online automatically due to strong affinities or cluster reconfiguration.

EXAMPLE 2–46 Specifying That a Critical Service Has Precedence

This example shows the commands for modifying the noncritical resource groups `ncrg1` and `ncrg2` to ensure that the critical resource group `mcdbrg` has precedence over these resource groups. This example assumes that resource groups `mcdbrg`, `ncrg1`, and `ncrg2` exist.

```
# clresourcegroup set -p RG_affinities=--mcdbrg ncrg1 ncrg2
```

Delegating the Failover or Switchover of a Resource Group

The source resource group of a strong positive affinity cannot fail over or be switched over to a node where the target of the affinity is not running. If you require the source resource group of a strong positive affinity to be allowed to fail over or be switched over, you must delegate the failover to the target resource group. When the target of the affinity fails over, the source of the affinity is forced to fail over with the target.

Note – You might need to switch over the source resource group of a strong positive affinity that is specified by the ++ operator. In this situation, switch over the target of the affinity and the source of the affinity at the same time.

To delegate failover or switchover of a resource group to another resource group, declare on the resource group a strong positive affinity with failover delegation for the other resource group.

```
# clresourcegroup set|create source-rg -p RG_affinities=+++target-rg
```

source-rg
Specifies the resource group that is delegating failover or switchover. This resource group is the resource group *on* which you are declaring a strong positive affinity with failover delegation for another resource group.

`-p RG_affinities=+++`*target-rg*
Specifies the resource group to which *source-rg* delegates failover or switchover. This resource group is the resource group *for* which you are declaring a strong positive affinity with failover delegation.

A resource group may declare a strong positive affinity with failover delegation for at most one resource group. However, a given resource group may be the target of strong positive affinities with failover delegation that are declared by any number of other resource groups.

A strong positive affinity with failover delegation is not fully symmetric. The target can come online while the source remains offline. However, if the target is offline, the source cannot come online.

If the target declares a strong positive affinity with failover delegation for a third resource group, failover or switchover is further delegated to the third resource group. The third resource group performs the failover or switchover, forcing the other resource groups to fail over or be switched over also.

EXAMPLE 2–47 Delegating the Failover or Switchover of a Resource Group

This example shows the command for modifying resource group `rg1` to declare a strong positive affinity with failover delegation for resource group `rg2`. As a result of this affinity relationship, `rg1` delegates failover or switchover to `rg2`. This example assumes that both resource groups exist.

```
# clresourcegroup set -p RG_affinities=+++rg2 rg1
```

Combining Affinities Between Resource Groups

You can create more complex behaviors by combining multiple affinities. For example, the state of an application might be recorded by a related replica server. The node selection requirements for this example are as follows:

- The replica server must run on a different node from the application.
- If the application fails over from its current node, the application should fail over to the node where the replica server is running.
- If the application fails over to the node where the replica server is running, the replica server must fail over to a different node. If no other node is available, the replica server must go offline.

You can satisfy these requirements by configuring resource groups for the application and the replica server as follows:

- The resource group that contains the application declares a weak positive affinity for the resource group that contains the replica server.
- The resource group that contains the replica server declares a strong negative affinity for the resource group that contains the application.

EXAMPLE 2–48 Combining Affinities Between Resource Groups

This example shows the commands for combining affinities between the following resource groups.

- Resource group `app-rg` represents an application whose state is tracked by a replica server.
- Resource group `rep-rg` represents the replica server.

In this example, the resource groups declare affinities as follows:

- Resource group `app-rg` declares a weak positive affinity for resource group `rep-rg`.
- Resource group `rep-rg` declares a strong negative affinity for resource group `app-rg`.

This example assumes that both resource groups exist.

```
# clresourcegroup set -p RG_affinities=+rep-rg app-rg
# clresourcegroup set -p RG_affinities=--app-rg rep-rg
```

Zone Cluster Resource Group Affinities

The cluster administrator can specify affinities between a resource group in a zone cluster and another resource group in a zone cluster or a resource group on the global cluster.

You can use the following command to specify the affinity between resource groups in zone clusters.

```
# clresourcegroup set -p RG_affinities=affinity-typetarget-zc:target-rg  source-zc:source-rg
```

The resource group affinity types in a zone cluster can be one of the following:

- + (weak positive)
- ++ (strong positive)
- - (weak negative)
- - - (strong negative)

Note – The affinity type +++ is not supported for zone clusters in this release.

EXAMPLE 2–49 Specifying a Strong Positive Affinity Between Resource Groups in Zone Clusters

This example shows the command for specifying a strong positive affinity between resource groups in zone clusters.

The resource group RG1 in a zone cluster ZC1 declares a strong positive affinity for a resource group RG2 in a zone cluster ZC2.

If you need to specify a strong positive affinity between a resource group RG1 in a zone cluster ZC1 and a resource group RG2 in another zone cluster ZC2, use the following command:

```
# clresourcegroup set -p RG_affinities=++ZC2:RG2 ZC1:RG1
```

EXAMPLE 2–50 Specifying a Strong Negative Affinity Between a Resource Group in a Zone Cluster and a Resource Group in the Global Cluster

This example shows the command for specifying a strong negative affinity between resource groups in zone clusters. If you need to specify a strong negative affinity between a resource group RG1 in a zone cluster ZC1 and a resource group RG2 in the global cluster, use the following command:

```
# clresourcegroup set -p RG_affinities=--global:RG2 ZC1:RG1
```

Replicating and Upgrading Configuration Data for Resource Groups, Resource Types, and Resources

If you require identical resource configuration data on two clusters, you can replicate the data to the second cluster to save the laborious task of setting it up again. Use scsnapshot to propagate the resource configuration information from one cluster to another cluster. To save effort, ensure that your resource-related configuration is stable and you do not need to make any major changes to the resource configuration, before copying the information to a second cluster.

Configuration data for resource groups, resource types, and resources can be retrieved from the Cluster Configuration Repository (CCR) and formatted as a shell script. The script can be used to perform the following tasks:

- Replicate configuration data on a cluster that does not have configured resource groups, resource types, or resources
- Upgrade configuration data on a cluster that has configured resource groups, resource types, and resources

The `scsnapshot` tool retrieves configuration data that is stored in the CCR. Other configuration data are ignored. The `scsnapshot` tool ignores the dynamic state of different resource groups, resource types, and resources.

▼ How to Replicate Configuration Data on a Cluster Without Configured Resource Groups, Resource Types, and Resources

This procedure replicates configuration data on a cluster that does not have configured resource groups, resource types, and resources. In this procedure, a copy of the configuration data is taken from one cluster and used to generate the configuration data on another cluster.

1 **Using the system administrator role, log in to any node in the cluster from which you want to copy the configuration data.**

For example, `node1`.

The system administrator role gives you the following role-based access control (RBAC) rights:

- `solaris.cluster.resource.read`
- `solaris.cluster.resource.modify`

2 **Retrieve the configuration data from the cluster.**

```
node1 % scsnapshot -s scriptfile
```

The `scsnapshot` tool generates a script called *scriptfile*. For more information about using the `scsnapshot` tool, see the `scsnapshot(1M)` man page.

3 **Edit the script to adapt it to the specific features of the cluster where you want to replicate the configuration data.**

For example, you might have to change the IP addresses and host names that are listed in the script.

4. **Launch the script from any node in the cluster where you want to replicate the configuration data.**

   The script compares the characteristics of the local cluster to the cluster where the script was generated. If the characteristics are not the same, the script writes an error and ends. A message asks whether you want to rerun the script, using the `-f` option. The `-f` option forces the script to run, despite any difference in characteristics. If you use the `-f` option, ensure that you do not create inconsistencies in your cluster.

   The script 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 running the script again.

▼ How to Upgrade Configuration Data on a Cluster With Configured Resource Groups, Resource Types, and Resources

This procedure upgrades configuration data on a cluster that already has configured resource groups, resource types, and resources. This procedure can also be used to generate a configuration template for resource groups, resource types, and resources.

In this procedure, the configuration data on `cluster1` is upgraded to match the configuration data on `cluster2`.

1. **Using the system administrator role, log on to any node in `cluster1`.**

   For example, `node1`.

   The system administrator role gives you the following RBAC rights:

   - `solaris.cluster.resource.read`
   - `solaris.cluster.resource.modify`

2. **Retrieve the configuration data from the cluster by using the image file option of the `scsnapshot` tool:**

   ```
   node1% scsnapshot -s scriptfile1 -o imagefile1
   ```

   When run on `node1`, the `scsnapshot` tool generates a script that is called *scriptfile1*. The script stores configuration data for the resource groups, resource types, and resources in an image file that is called *imagefile1*. For more information about using the `scsnapshot` tool, see the `scsnapshot`(1M) man page.

3. **Repeat Step 1 through Step 2 on a node in `cluster2`:**

   ```
   node2 % scsnapshot -s scriptfile2 -o imagefile2
   ```

4 **On `node1`, generate a script to upgrade the configuration data on `cluster1` with configuration data from `cluster2`:**

```
node1 % scsnapshot -s scriptfile3 imagefile1 imagefile2
```

This step uses the image files that you generated in Step 2 and Step 3, and generates a new script that is called *scriptfile3*.

5 **Edit the script that you generated in Step 4 to adapt it to the specific features of the `cluster1`, and to remove data specific to `cluster2`.**

6 **From `node1`, launch the script to upgrade the configuration data.**

The script compares the characteristics of the local cluster to the cluster where the script was generated. If the characteristics are not the same, the script writes an error and ends. A message asks whether you want to rerun the script, using the `-f` option. The `-f` option forces the script to run, despite any difference in characteristics. If you use the `-f` option, ensure that you do not create inconsistencies in your cluster.

The script 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 running the script again.

Enabling Solaris SMF Services to Run With Sun Cluster

The Service Management Facility (SMF) enables you to automatically start and restart SMF services, during a node boot or service failure. SMF facilitates some degree of high availability to the SMF services on a single host. This feature is similar to the Sun Cluster Resource Group Manager (RGM), which facilitates high availability and scalability for cluster applications. SMF services and RGM features are complementary to each other.

Sun Cluster includes three new SMF proxy resource types that can be used to enable SMF services to run with Sun Cluster in a failover, multi-master, or scalable configuration. The following are the proxy resource types:

- `SUNW.Proxy_SMF_failover`
- `SUNW.Proxy_SMF_multimaster`
- `SUNW.Proxy_SMF_scalable`

The SMF proxy resource types enables you to encapsulate a set of interrelated SMF services into a single resource, *SMF proxy resource* to be managed by Sun Cluster. In this feature, SMF manages the availability of SMF services on a single node. Sun Cluster provides cluster-wide high availability and scalability of the SMF services.

You can use the SMF proxy resource types to integrate your own SMF controlled services into Sun Cluster so that these services have cluster-wide service availability without you rewriting callback methods or service manifest. After you integrate the SMF service into the SMF proxy resource, the SMF service is no longer managed by the default restarter. The restarter that is delegated by Sun Cluster manages the SMF service.

SMF proxy resources are identical to other resources, with no restriction on their usage. For example, an SMF proxy resource can be grouped with other resources into a resource group. SMF proxy resources can be created and managed the same way as other resources. An SMF proxy resource differs from other resources in one way. When you create a resource of any of the SMF proxy resource types, you need to specify the extension property `Proxied_service_instances`. You must include information about the SMF services to be proxied by the SMF resource. The extension property's 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`, `path of 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>,</var/svc/manifest/system/clus
ter/restarter_svc_test_1.xml>
```

```
<svc:/system/cluster/restarter_svc_test_2:default>,</var/svc/manifest/system/clus
ter/restarter_svc_test_2.xml>
```

The services that are encapsulated under an SMF proxy resource can reside in the global cluster or global-cluster non-voting node. However, all the services under the same proxy resource must be in the same zone.

Caution – Do not use SMF svcadm for disabling or enabling SMF services that are encapsulated in a proxy resource. Do not change the properties of the SMF services (in the SMF repository) that are encapsulated in a proxy resource.

- "Encapsulating an SMF Service Into a Failover Proxy Resource Configuration" on page 148
- "Encapsulating an SMF Service Into a Multi-Master Proxy Resource Configuration" on page 151
- "Encapsulating an SMF Service Into a Scalable Proxy Resource Configuration" on page 154

▼ Encapsulating an SMF Service Into a Failover Proxy Resource Configuration

For information about failover configuration, see "Creating a Resource Group" on page 41

Note – Perform this procedure from any cluster node.

---

1 **On a cluster member, become superuser or assume a role that provides `solaris.cluster.modify` RBAC authorization.**

2 **Register the proxy SMF failover resource type.**

```
# clresourcetype register -f\
/opt/SUNWscsmf/etc/SUNW.Proxy_SMF_failover SUNW.Proxy_SMF_failover
```

3 **Verify that the proxy resource type has been registered.**

```
# clresourcetype show
```

4 **Create the SMF failover resource group.**

```
# clresourcegroup create [-n node-zone-list] resource-group
```

`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes that can master this resource group. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the resource group is configured on all the global-cluster voting nodes. **Note –** To achieve highest availability, specify global-cluster non-voting nodes on different nodes in the node list of an SMF failover resource group instead of different global-cluster non-voting nodes on the same node.
resource-group	Specifies your choice of the name of the scalable resource group to add. This name must begin with an ASCII character.

5 **Verify that the SMF resource group has been created.**

```
# clresourcegroup status resource-group
```

6 **Add an SMF failover application resource to the resource group.**

```
# clresourcegroup create -g resource-group -t SUNW.Proxy_SMF_failover \
[-p "extension-property[{node-specifier}]"=value, ...] [-p standard-property=value, ...] resource
```

The resource is created in the enabled state.

7 **Verify that the SMF failover application resource has been added and validated.**

```
# clresource show resource
```

8 **Bring the failover resource group online.**

```
# clresourcegroup online -M +
```

Note – If you use the `clresource status` command to view the state of the SMF proxy resource type, the status is displayed as `online but not monitored`. This is not an error message. The SMF proxy resource is enabled and running and this status message is displayed because there is no monitoring support provided for the resources of SMF proxy resource type.

Example 2–51 Registering an SMF Proxy Failover Resource Type

The following example registers the `SUNW.Proxy_SMF_failover` resource type.

```
# clresourcetype register SUNW.Proxy_SMF_failover
# clresourcetype show SUNW.Proxy_SMF_failover

Resource Type:              SUNW.Proxy_SMF_failover
RT_description:             Resource type for proxying failover SMF services
RT_version:                 3.2
API_version:                6
RT_basedir:                 /opt/SUNWscsmf/bin
Single_instance:            False
Proxy:                      False
Init_nodes:                 All potential masters
Installed_nodes:            <All>
Failover:                   True
Pkglist:                    SUNWscsmf
RT_system:                  False
Global_zone:                   False
```

Example 2–52 Adding an SMF Proxy Failover Application Resource to a Resource Group

This example shows the addition of a proxy resource type, `SUN.Proxy_SMF_failover` to a resource group `resource-group-1`.

```
# clresource create -g resource-group-1 -t SUNW.Proxy_SMF_failover
-x proxied_service_instances=/var/tmp/svslist.txt resource-1
# clresource show resource-1

=== Resources ===

  Resource:                                         resource-1
  Type:                                             SUNW.Proxy_SMF_failover
  Type_version:                                     3.2
  Group:                                            resource-group-1
```

```
R_description:
Resource_project_name:                      default
Enabled{phats1}:                            True
Monitored{phats1}:                          True
```

▼ Encapsulating an SMF Service Into a Multi-Master Proxy Resource Configuration

1. **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2. **Register the SMF proxy multi-master resource type.**

```
# clresourcetype register -f\
/opt/SUNWscsmf/etc/SUNW.Proxy_SMF_multimaster SUNW.Proxy_SMF_multimaster
```

3. **Create the SMF multi-master resource group.**

```
# clresourcegroup create\-p Maximum_primaries=m\-p Desired_primaries=n\
[-n node-zone-list]\
resource-group
```

`-p Maximum_primaries=`*m*	Specifies the maximum number of active primaries for this resource group.
`-p Desired_primaries=`*n*	Specifies the number of active primaries on which the resource group should attempt to start.
`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes in which this resource group is to be available. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the node name and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the resource group is configured on the global-cluster voting nodes.
resource-group	Specifies your choice of the name of the scalable resource group to add. This name must begin with an ASCII character.

4. **Verify that the SMF proxy multi-master resource group has been created.**

```
# clresourcegroup show resource-group
```

5 **Add an SMF proxy multi-master resource to the resource group.**

```
# clresource create -g resource-group -t SUNW.Proxy_SMF_multimaster\
[-p "extension-property[{node-specifier}]"=value, ...] [-p standard-property=value, ...] resource
```

-g *resource-group*
Specifies the name of a scalable service resource group that you previously created.

-p "*extension-property*[**{***node-specifier***}**]"=*value*, …
Specifies a comma-separated list of extension properties that you are setting for the resource. The extension properties that you can set depend on the resource type. To determine which extension properties to set, see the documentation for the resource type.

node-specifier is an *optional* qualifier to the -p and -x options. This qualifier indicates that the extension property or properties on *only* the specified node or nodes are to be set when the resource is created. The specified extension properties on other nodes in the cluster are not set. If you do not include *node-specifier*, the specified extension properties on all nodes in the cluster are set. You can specify a node name or a node identifier for *node-specifier*. Examples of the syntax of *node-specifier* include the following:

```
-p "myprop{phys-schost-1}"
```

The braces ({}) indicate that you want to set the specified extension property on only node phys-schost-1. For most shells, the double quotation marks (“) are required.

You can also use the following syntax to set an extension property in two different global-cluster non-voting nodes on two different nodes:

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

-p *standard-property*=*value*, …
Specifies a comma-separated list of standard properties that you are setting for the resource. The standard properties that you can set depend on the resource type. For scalable services, you typically set the Port_list, Load_balancing_weights, and Load_balancing_policy properties. To determine which standard properties to set, see the documentation for the resource type and Appendix B, “Standard Properties.”

resource
Specifies your choice of the name of the resource to add.

The resource is created in the enabled state.

6 **Verify that the SMF proxy multi-master application resource has been added and validated.**

```
# clresource show resource
```

7 **Bring the multi-master resource group online.**

```
# clresourcegroup online -M +
```

Note – If you use the `clresource status` command to view the state of the SMF proxy resource type, the status is displayed as `online but not monitored`. This is not an error message. The SMF proxy resource is enabled and running and this status message is displayed because there is no monitoring support provided for the resources of SMF proxy resource type.

Example 2–53 Registering an SMF Proxy Multi-Master Resource Type

The following example registers the `SUNW.Proxy_SMF_multimaster` resource type.

```
# clresourcetype register SUNW.Proxy_SMF_multimaster
# clresourcetype show SUNW.Proxy_SMF_multimaster

Resource Type:            SUNW.Proxy_SMF_multimaster
RT_description:           Resource type for proxying multimastered SMF services
RT_version:               3.2
API_version:              6
RT_basedir:               /opt/SUNWscsmf/bin
Single_instance:          False
Proxy:                    False
Init_nodes:               All potential masters
Installed_nodes:          <All>
Failover:                  True
Pkglist:                   SUNWscsmf
RT_system:                 False
Global_zone:                   False
```

Example 2–54 Creating and Adding an SMF Proxy Multi-Master Application Resource to a Resource Group

This example shows the creation and addition of a multi-master proxy resource type `SUN.Proxy_SMF_multimaster` to a resource group `resource-group-1`.

```
# clresourcegroup create\
-p Maximum_primaries=2\
-p Desired_primaries=2\
-n phys-schost-1, phys-schost-2\
resource-group-1
# clresourcegroup show resource-group-1

=== Resource Groups and Resources ===

Resource Group:                         resource-group-1
RG_description:                         <NULL>
RG_mode:                                multimastered
RG_state:                               Unmanaged
```

```
RG_project_name:                       default
RG_affinities:                         <NULL>
Auto_start_on_new_cluster:             True
Failback:                              False
Nodelist:                              phys-schost-1 phys-schost-2
Maximum_primaries:                      2
Desired_primaries:                      2
Implicit_network_dependencies:         True
Global_resources_used:                 <All>
Pingpong_interval:                      3600
Pathprefix:                            <NULL>
RG_System:                             False
Suspend_automatic_recovery:                      False

# clresource create -g resource-group-1 -t SUNW.Proxy_SMF_multimaster
-x proxied_service_instances=/var/tmp/svslist.txt resource-1
# clresource show resource-1

=== Resources ===

  Resource:                              resource-1
  Type:                                 SUNW.Proxy_SMF_multimaster
  Type_version:                         3.2
  Group:                                resource-group-1
  R_description:
  Resource_project_name:                default
  Enabled{phats1}:                      True
  Monitored{phats1}:                    True
```

▼ Encapsulating an SMF Service Into a Scalable Proxy Resource Configuration

For information about scalable configuration, see "How to Create a Scalable Resource Group" on page 43

Note – Perform this procedure from any cluster node.

1 **On a cluster member, become superuser or assume a role that provides** `solaris.cluster.modify` **RBAC authorization.**

2 **Register the SMF proxy scalable resource type.**

```
# clresourcetype register -f\
/opt/SUNWscsmf/etc/SUNW.Proxy_SMF_scalable SUNW.Proxy_SMF_scalable
```

3 **Create the SMF failover resource group that holds the shared address that the scalable resource group is to use. See "How to Create a Failover Resource Group" on page 41 to create the failover resource group.**

4 **Create the SMF proxy scalable resource group.**

```
# clresourcegroup create\-p Maximum_primaries=m\-p Desired_primaries=n\
-p RG_dependencies=depend-resource-group\
[-n node-zone-list]\
resource-group
```

`-p Maximum_primaries=`*m*	Specifies the maximum number of active primaries for this resource group.
`-p Desired_primaries=`*n*	Specifies the number of active primaries on which the resource group should attempt to start.
`-p RG_dependencies=`*depend-resource-group*	Identifies the resource group that contains the shared address-resource on which the resource group that is being created depends.
`-n` *node-zone-list*	Specifies a comma-separated, ordered list of nodes in which this resource group is to be available. The format of each entry in the list is *node*`:`*zone*. In this format, *node* specifies the global-cluster voting node and *zone* specifies the name of a global-cluster non-voting node. To specify the global-cluster voting node, or to specify a node without global-cluster non-voting nodes, specify only *node*. This list is optional. If you omit this list, the resource group is created on all nodes in the cluster. The node list of the scalable resource can contain the same list or a subset of *nodename:zonename* pairs as the node list of the shared address resource
resource-group	Specifies your choice of the name of the scalable resource group to add. This name must begin with an ASCII character.

5 **Verify that the scalable resource group has been created.**

```
# clresourcegroup show resource-group
```

6 **Add an SMF proxy scalable resource to the resource group.**

```
# clresource create-g resource-group -t SUNW.Proxy_SMF_scalable \
-p Network_resources_used=network-resource[,network-resource...] \
-p Scalable=True
[-p "extension-property[{node-specifier}]"=value, ...] [-p standard-property=value, ...] resource
```

`-g` *resource-group*
Specifies the name of a scalable service resource group that you previously created.

`-p Network_resources_used=` *network-resource*[,*network-resource*...]
Specifies the list of network resources (shared addresses) on which this resource depends.

`-p Scalable=True`
Specifies that this resource is scalable.

`-p` "*extension-property*[**{***node-specifier***}**]"**=***value*, …
Specifies that you are setting extension properties for the resource. To determine which extension properties to set, see the documentation for the resource type.

node-specifier is an *optional* qualifier to the `-p` and `-x` options. This qualifier indicates that the extension property or properties on *only* the specified node or nodes are to be set when the resource is created. The specified extension properties on other nodes in the cluster are not set. If you do not include *node-specifier*, the specified extension properties on all nodes in the cluster are set. You can specify a node name or a node identifier for *node-specifier*. Examples of the syntax of *node-specifier* include the following:

```
-p "myprop{phys-schost-1}"
```

The braces ({}) indicate that you want to set the specified extension property on only node `phys-schost-1`. For most shells, the double quotation marks (“) are required.

You can also use the following syntax to set an extension property in two different global-cluster non-voting nodes on two different global-cluster voting nodes:

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

`-p` *standard-property*=*value*, …
Specifies a comma-separated list of standard properties that you are setting for the resource. The standard properties that you can set depend on the resource type. For scalable services, you typically set the `Port_list`, `Load_balancing_weights`, and `Load_balancing_policy` properties. To determine which standard properties to set, see the documentation for the resource type and Appendix B, “Standard Properties.”

resource
Specifies your choice of the name of the resource to add.

The resource is created in the enabled state.

7 **Verify that the SMF proxy scalable application resource has been added and validated.**

```
# clresource show resource
```

8 **Bring the SMF proxy scalable resource group online.**

```
# clresourcegroup online -M +
```

Note – If you use the `clresource status` command to view the state of the SMF proxy resource type, the status is displayed as `online but not monitored`. This is not an error message. The SMF proxy resource is enabled and running and this status message is displayed because there is no monitoring support provided for the resources of SMF proxy resource type.

Example 2–55 Registering an SMF Proxy Scalable Resource Type

The following example registers the `SUNW.Proxy_SMF_scalable` resource type.

```
# clresourcetype register SUNW.Proxy_SMF_scalable
# clresourcetype show SUNW.Proxy_SMF_scalable

Resource Type:           SUNW.Proxy_SMF_scalable
RT_description:          Resource type for proxying scalable SMF services
RT_version:              3.2
API_version:             6
RT_basedir:              /opt/SUNWscsmf/bin
Single_instance:          False
Proxy:                    False
Init_nodes:               All potential masters
Installed_nodes:          <All>
Failover:                 True
Pkglist:                  SUNWscsmf
RT_system:                 False
Global_zone:                  False
```

Example 2–56 Creating and Adding an SMF Proxy Scalable Application Resource to a Resource Group

This example shows the creation and addition of a scalable proxy resource type `SUN.Proxy_SMF_scalalble` to a resource group `resource-group-1`.

```
# clresourcegroup create\
-p Maximum_primaries=2\
-p Desired_primaries=2\
-p RG_dependencies=resource-group-2\
-n phys-schost-1, phys-schost-2\
resource-group-1
# clresourcegroup show resource-group-1

=== Resource Groups and Resources ===
Resource Group:                      resource-group-1
RG_description:                     <NULL>
RG_mode:                             Scalable
RG_state:                            Unmanaged
RG_project_name:                     default
RG_affinities:                       <NULL>
Auto_start_on_new_cluster:           True
Failback:                            False
Nodelist:                            phys-schost-1 phys-schost-2
Maximum_primaries:                   2
Desired_primaries:                   2
RG_dependencies:                     resource-group2
Implicit_network_dependencies:       True
Global_resources_used:               <All>
Pingpong_interval:                   3600
Pathprefix:                          <NULL>
RG_System:                            False
Suspend_automatic_recovery:           False

# clresource create -g resource-group-1 -t SUNW.Proxy_SMF_scalable
-x proxied_service_instances=/var/tmp/svslist.txt resource-1
# clresource show resource-1

=== Resources ===

  Resource:                            resource-1
  Type:                                SUNW.Proxy_SMF_scalable
  Type_version:                        3.2
  Group:                               resource-group-1
  R_description:
  Resource_project_name:               default
  Enabled{phats1}:                     True
  Monitored{phats1}:                   True
```

Tuning Fault Monitors for Sun Cluster Data Services

Each data service that is supplied with the Sun Cluster product has a built-in fault monitor. The fault monitor performs the following functions:

- Detecting the unexpected termination of processes for the data service server
- Checking the health of the data service

The fault monitor is contained in the resource that represents the application for which the data service was written. You create this resource when you register and configure the data service. For more information, see the documentation for the data service.

System properties and extension properties of this resource control the behavior of the fault monitor. The default values of these properties determine the preset behavior of the fault monitor. The preset behavior should be suitable for most Sun Cluster installations. Therefore, you should tune a fault monitor *only* if you need to modify this preset behavior.

Tuning a fault monitor involves the following tasks:

- Setting the interval between fault monitor probes
- Setting the timeout for fault monitor probes
- Defining the criteria for persistent faults
- Specifying the failover behavior of a resource

Perform these tasks when you register and configure the data service. For more information, see the documentation for the data service.

Note – A resource's fault monitor is started when you bring online the resource group that contains the resource. You do not need to start the fault monitor explicitly.

Setting the Interval Between Fault Monitor Probes

To determine whether a resource is operating correctly, the fault monitor probes this resource periodically. The interval between fault monitor probes affects the availability of the resource and the performance of your system as follows:

- The interval between fault monitor probes affects the length of time that is required to detect a fault and respond to the fault. Therefore, if you decrease the interval between fault monitor probes, the time that is required to detect a fault and respond to the fault is also decreased. This decrease enhances the availability of the resource.
- Each fault monitor probe consumes system resources such as processor cycles and memory. Therefore, if you decrease the interval between fault monitor probes, the performance of the system is degraded.

The optimum interval between fault monitor probes also depends on the time that is required to respond to a fault in the resource. This time depends on how the complexity of the resource affects the time that is required for operations such as restarting the resource.

To set the interval between fault monitor probes, set the `Thorough_probe_interval` system property of the resource to the interval in seconds that you require.

Setting the Timeout for Fault Monitor Probes

The timeout for fault monitor probes specifies the length of time that a fault monitor waits for a response from a resource to a probe. If the fault monitor does not receive a response within this timeout, the fault monitor treats the resource as faulty. The time that a resource requires to respond to a fault monitor probe depends on the operations that the fault monitor performs to probe the resource. For information about operations that a data service's fault monitor performs to probe a resource, see the documentation for the data service.

The time that is required for a resource to respond also depends on factors that are unrelated to the fault monitor or the application, for example:

- System configuration
- Cluster configuration
- System load
- Amount of network traffic

To set the timeout for fault monitor probes, set the `Probe_timeout` extension property of the resource to the timeout in seconds that you require.

Defining the Criteria for Persistent Faults

To minimize the disruption that transient faults in a resource cause, a fault monitor restarts the resource in response to such faults. For persistent faults, more disruptive action than restarting the resource is required:

- For a failover resource, the fault monitor fails over the resource to another node.
- For a scalable resource, the fault monitor takes the resource offline.

A fault monitor treats a fault as persistent if the number of complete failures of a resource exceeds a specified threshold within a specified retry interval. Defining the criteria for persistent faults enables you to set the threshold and the retry interval to accommodate the performance characteristics of your cluster and your availability requirements.

Complete Failures and Partial Failures of a Resource

A fault monitor treats some faults as a *complete failure* of a resource. A complete failure typically causes a complete loss of service. The following failures are examples of a complete failure:

- Unexpected termination of the process for a data service server
- Inability of a fault monitor to connect to a data service server

A complete failure causes the fault monitor to increase by 1 the count of complete failures in the retry interval.

A fault monitor treats other faults as a *partial failure* of a resource. A partial failure is less serious than a complete failure, and typically causes a degradation of service, but not a complete loss of service. An example of a partial failure is an incomplete response from a data service server before a fault monitor probe is timed out.

A partial failure causes the fault monitor to increase by a fractional amount the count of complete failures in the retry interval. Partial failures are still accumulated over the retry interval.

The following characteristics of partial failures depend on the data service:

- The types of faults that the fault monitor treats as partial failure
- The fractional amount that each partial failure adds to the count of complete failures

For information about faults that a data service's fault monitor detects, see the documentation for the data service.

Dependencies of the Threshold and the Retry Interval on Other Properties

The maximum length of time that is required for a single restart of a faulty resource is the sum of the values of the following properties:

- `Thorough_probe_interval` system property
- `Probe_timeout` extension property

To ensure that you allow enough time for the threshold to be reached within the retry interval, use the following expression to calculate values for the retry interval and the threshold:

retry_interval `>= 2 x` *threshold* `× (`*thorough_probe_interval* + *probe_timeout*`)`

The factor of 2 accounts for partial probe failures that do not immediately cause the resource to be failed over or taken offline.

System Properties for Setting the Threshold and the Retry Interval

To set the threshold and the retry interval, set the following system properties of the resource:

- To set the threshold, set the `Retry_count` system property to the maximum allowed number of complete failures.
- To set the retry interval, set the `Retry_interval` system property to the interval in seconds that you require.

Specifying the Failover Behavior of a Resource

The failover behavior of a resource determines how the RGM responds to the following faults:

- Failure of the resource to start
- Failure of the resource to stop
- Failure of the resource's fault monitor to stop

To specify the failover behavior of a resource, set the `Failover_mode` system property of the resource. For information about the possible values of this property, see the description of the `Failover_mode` system property in "Resource Properties" on page 185.

◆ ◆ ◆ APPENDIX A

Sun Cluster Object-Oriented Commands

This appendix introduces the object-oriented commands, their short forms, and their subcommands.

Object-Oriented Command Names and Aliases

In addition to their longer and more descriptive forms, many Sun Cluster commands also have a short form, or alias, that significantly reduces the amount you must type. The following table lists the commands and their shorter aliases.

TABLE A–1 Object-Oriented Commands and Aliases (Short Names)

Full Command	Alias	Purpose
`claccess`	none	Manage Sun Cluster access policies
`cldevice`	`cldev`	Manage Sun Cluster devices
`cldevicegroup`	`cldg`	Manage Sun Cluster device groups
`clinterconnect`	`clintr`	Manage the Sun Cluster interconnect
`clnasdevice`	`clnas`	Manage access to NAS devices for Sun Cluster
`clnode`	none	Manage Sun Cluster nodes
`clquorum`	`clq`	Manage Sun Cluster quorum
`clquorumserver`	`clqs`	Configure and manage quorum server processes on the quorum server host
`clreslogicalhostname`	`clrslh`	Manage Sun Cluster resources for logical host names
`clresource`	`clrs`	Manage resources for Sun Cluster data services

TABLE A–1 Object-Oriented Commands and Aliases (Short Names) *(Continued)*

Full Command	Alias	Purpose
`clresourcegroup`	`clrg`	Manage resource groups for Sun Cluster data services
`clresourcetype`	`clrt`	Manage resource types for Sun Cluster data services
`clrssharedaddress`	`clrssa`	Manage Sun Cluster resources for shared addresses
`clsetup`	none	Configure Sun Cluster interactively. This command has no subcommands.
`clsnmphost`	none	Administer Sun Cluster SNMP hosts
`clsnmpmib`	none	Administer the Sun Cluster SNMP MIB
`clsnmpuser`	none	Administer Sun Cluster SNMP users
`cltelemetryattribute`	`clta`	Configure system resource monitoring.
`cluster`	none	Manage the global configuration and status of Sun Cluster
`clvxvm`	none	Configure Veritas Volume Manager for Sun Cluster
`clzonecluster`	`clzc`	Manage zone clusters

Object-Oriented Command Set Overview

The following tables list the commands in the object-oriented command set and the subcommands available with each command.

TABLE A–2 `claccess`: Manage Sun Cluster Access Policies for Nodes

Subcommand	Purpose
`allow`	Allows the specified machine or machines access to the cluster configuration.
`allow-all`	Allows all nodes access to the cluster configuration.
`deny`	Denies the specified machine or machines access to the cluster configuration.
`deny-all`	Denies all nodes access to the cluster configuration.
`list`	Displays the names of the machines that have access to the cluster configuration.
`set`	Sets the authentication protocol to the value that you specify with the `-a` option.
`show`	Displays the names of the machines that have access to the cluster configuration.

TABLE A–3 `cldevice`, `cldev`: Manage Sun Cluster Devices

Subcommand	Purpose
`check`	Performs a consistency check to compare the kernel representation of the devices against the physical devices.
`clear`	Removes all DID references to underlying devices that are detached from the current node.
`combine`	Combines the specified DID instance with a new destination instance.
`export`	Exports configuration information for a cluster device.
`list`	Displays all device paths.
`monitor`	Turns on monitoring for the specified disk paths.
`populate`	Populates the global-devices namespace.
`refresh`	Updates the device configuration information that is based on the current device trees on a cluster node.
`rename`	Moves the specified DID instance to a new DID instance.
`repair`	Performs a repair procedure on the specified device instances.
`replicate`	Configures DID devices for use with controller-based replication.
`set`	Sets the properties of the specified device.
`show`	Displays a configuration report for all specified device paths.
`status`	Displays the status of the disk paths that are specified as operands to the command.
`unmonitor`	Turns off monitoring for the disk paths that are specified as operands to the command.

TABLE A–4 `cldevicegroup`, `cldg`: Manage Sun Cluster Device Groups

Subcommand	Purpose
`add-device`	Adds new member disk devices to an existing raw-disk device group.
`add-node`	Adds new nodes to an existing device group.
`create`	Creates a new device group.
`delete`	Deletes device groups.
`disable`	Disables offline device groups.
`enable`	Enables device groups.
`export`	Exports the device-group configuration information.
`list`	Displays a list of device groups.

TABLE A–4 `cldevicegroup`, `cldg`: Manage Sun Cluster Device Groups *(Continued)*

Subcommand	Purpose
`offline`	Takes device groups offline.
`online`	Brings device groups online on a predesignated node.
`remove-device`	Removes member disk devices from a raw-disk device group.
`remove-node`	Removes nodes from existing device groups.
`set`	Sets attributes that are associated with a device group.
`show`	Generates a configuration report for device groups.
`status`	Generates a status report for device groups.
`switch`	Transfers device groups from one primary node in a Sun Cluster configuration to another node.
`sync`	Synchronizes device-group information with the clustering software.

TABLE A–5 `clinterconnect, clintr:` Manage the Sun Cluster Interconnect

Subcommand	Purpose
`add`	Adds the new cluster interconnect components that are specified as operands to the command.
`disable`	Disables the interconnect components that are specified as operands to the command.
`enable`	Enables the interconnect components that are specified as operands to the command.
`export`	Exports the cluster interconnect configuration information.
`remove`	Removes the cluster interconnect components that are supplied as operands to the command.
`show`	Displays the configuration of interconnect components.
`status`	Displays the status of the interconnect paths.

TABLE A–6 `clnasdevice`, `clnas`: Manage Access to NAS Devices for Sun Cluster

Subcommand	Purpose
`add`	Adds a NAS device to the Sun Cluster configuration.
`add-dir`	Adds the specified directories of an already configured NAS device to the cluster configuration.
`export`	Exports the cluster NAS device configuration information.
`list`	Displays the NAS devices configured in the cluster.

TABLE A–6 `clnasdevice`, `clnas`: Manage Access to NAS Devices for Sun Cluster *(Continued)*

Subcommand	Purpose
`remove`	Removes the specified NAS device or devices from the Sun Cluster configuration.
`remove-dir`	Removes the specified NAS directory or directories from the Sun Cluster configuration.
`set`	Sets specified properties of a specific NAS device.
`show`	Displays configuration information for NAS devices in the cluster.

TABLE A–7 `clnode`: Manage Sun Cluster Nodes

Subcommand	Purpose
`add`	Configures and adds a node to the cluster.
`add-farm`	Adds a farm node to a cluster.
`clear`	Removes a node from the Sun Cluster software configuration.
`evacuate`	Attempts to switch over all resource groups and device groups from the specified node to a new set of primary nodes.
`export`	Exports the node or farm configuration information to a file or to the standard output `(stdout).`
`list`	Displays the names of nodes that are configured in the cluster or in the farm.
`remove`	Removes a node from the cluster.
`remove-farm`	Removes a farm node from a cluster.
`set`	Sets the properties that are associated with the node that you specify.
`show`	Displays the configuration of the specified node or nodes.
`show-rev`	Displays the names of and release information about the Sun Cluster packages that are installed on a node.
`status`	Displays the status of the node or nodes that you specify.

TABLE A–8 `clquorum`, `clq`: Manage Sun Cluster Quorum Configuration

Subcommand	Purpose
`add`	Adds the specified shared device as a quorum device.
`disable`	Puts a quorum device or node in the quorum maintenance state.
`enable`	Removes a quorum device or a node from the quorum maintenance state.
`export`	Exports the configuration information for the cluster quorum.
`list`	Displays the names of quorum devices that are configured in the cluster.

TABLE A–8 `clquorum`, `clq`: Manage Sun Cluster Quorum Configuration *(Continued)*

Subcommand	Purpose
`remove`	Removes the specified quorum device or devices from the Sun Cluster quorum configuration.
`reset`	Resets the entire quorum configuration to the default vote count settings.
`show`	Displays the properties of quorum devices.
`status`	Displays the status and vote counts of quorum devices.

TABLE A–9 `clquorumserver`, `clqs`: Manage Quorum Servers

Subcommand	Purpose
`clear`	Removes outdated cluster information from the quorum server.
`show`	Displays the configuration information about the quorum server.
`start`	Starts the quorum server process on the host machine.
`stop`	Stops the quorum server process.

TABLE A–10 `clreslogicalhostname`, `clrslh`: Manage Resources for Sun Cluster Logical Host Names

Subcommand	Purpose
`create`	Creates new logical hostname resources.
`delete`	Deletes logical hostname resources.
`disable`	Disables logical hostname resources.
`enable`	Enables logical hostname resources.
`export`	Exports logical hostname resource configuration,
`list`	Displays a list of the logical hostname resources.
`list-props`	Displays a list of the properties of the logical hostname resources.
`monitor`	Turns on monitoring for logical hostname resources.
`reset`	Clears an error flag that is associated with logical hostname resources.
`set`	Sets specified properties of the logical hostname resources.
`show`	Displays the configuration of logical hostname resources.
`status`	Displays the status of logical hostname resources.
`unmonitor`	Turns off monitoring for logical hostname resources.

TABLE A–11 `clresource`, `clrs`: Manage Resources for Sun Cluster Data Services

Subcommand	Purpose
`create`	Creates the resources that are specified as operands to the command.
`delete`	Deletes the resources that are specified as operands to the command.
`disable`	Disables resources.
`enable`	Enables resources.
`export`	Exports the cluster resource configuration.
`list`	Displays a list of cluster resources.
`list-props`	Displays a list of resource properties.
`monitor`	Turns on monitoring for resources.
`reset`	Clears error flags that are associated with cluster resources.
`set`	Sets resource properties.
`show`	Displays resource configuration.
`status`	Displays resource status.
`unmonitor`	Turns off resource monitoring.

TABLE A–12 `clresourcegroup`, `clrg`: Manage Resource Groups for Sun Cluster Data Services

Subcommand	Purpose
`add-node`	Adds a node to the end of the `Nodelist` property for a resource group.
`create`	Creates a new resource group.
`delete`	Deletes a resource group.
`evacuate`	Brings offline all resource groups on the nodes that you specify with the `-n` option.
`export`	Writes the configuration information for a resource group to a file or to the standard output (`stdout`).
`list`	Displays a list of resource groups.
`manage`	Brings a resource group that you specify to a managed state.
`offline`	Brings a resource group that you specify to an offline state.
`online`	Brings a resource group that you specify to an online state.
`quiesce`	Brings the specified resource group to a quiescent state.
`remaster`	Switches a resource group that you specify to its most preferred node.

TABLE A–12 `clresourcegroup`, `clrg`: Manage Resource Groups for Sun Cluster Data Services *(Continued)*

Subcommand	Purpose
`remove-node`	Removes a node from the `Nodelist` property of a resource group.
`restart`	Takes a resource group offline and then back online on the same set of primary nodes that originally hosted the resource group.
`resume`	Clears the suspended state of any suspended resource groups that you specify.
`set`	Sets the properties that are associated with the resource groups that you specify.
`show`	Generates a configuration report for resource groups that you specify.
`status`	Generates a status report for resource groups that you specify.
`suspend`	Suspends RGM control over all applications that are managed by a resource group that you specify.
`switch`	Changes the node, or set of nodes, that is mastering a resource group that you specify.
`unmanage`	Brings a resource group that you specify to an unmanaged state.

TABLE A–13 `clresourcetype`, `clrt`: Manage Resource Types for Sun Cluster Data Services

Subcommand	Purpose
`add-node`	Adds the specified nodes to the node list for resource types.
`export`	Exports the cluster resource-type configuration.
`list`	Displays a list of resource types.
`list-props`	Displays a list of the resource extension properties or resource type properties of resource types.
`register`	Registers resource types.
`remove-node`	Removes a node from the list of nodes for which the resource types in the operand list are registered.
`set`	Sets properties of resource types.
`show`	Displays configuration information about resource types that are registered in the cluster.
`unregister`	Unregisters resource types.

TABLE A–14 `clressharedaddress`, `clrssa`: Manage Sun Cluster Resources for Shared Addresses

Subcommand	Purpose
`create`	Creates shared address resources.

TABLE A–14 `clresssharedaddress`, `clrssa`: Manage Sun Cluster Resources for Shared Addresses *(Continued)*

Subcommand	Purpose
`delete`	Deletes shared address resources.
`disable`	Disables shared address resources.
`enable`	Enables shared address resources.
`export`	Exports shared address resource configuration.
`list`	Displays a list of shared address resources.
`list-props`	Displays a list of properties of shared address resources.
`monitor`	Turns on monitoring for shared address resources.
`reset`	Clears an error flag that is associated with shared address resources.
`set`	Sets specified properties of shared address resources.
`show`	Displays the configuration of shared address resources.
`status`	Displays the status of shared address resources.
`unmonitor`	Turns off monitoring for shared address resources.

TABLE A–15 `clsnmphost`: Administer the List of Sun Cluster SNMP Hosts

Subcommand	Purpose
`add`	Adds an SNMP host to the specified node configuration.
`export`	Exports the SNMP host information from the specified node.
`list`	Lists the SNMP hosts that are configured on the specified node.
`remove`	Removes an SNMP host from the node configuration.
`show`	Displays the SNMP host configuration information about the specified node.

TABLE A–16 `clsnmpmib`: Administer Sun Cluster SNMP MIB

Subcommand	Purpose
`disable`	Disables one or more of the cluster MIBs on the specified nodes.
`enable`	Enables one or more cluster MIBs on the specified node.
`export`	Exports the cluster MIB configuration information.
`list`	Displays a list of cluster MIBs on the specified nodes.
`set`	Sets the SNMP protocol setting that is used on one or more of the MIBs.

TABLE A–16 `clsnmpmib`: Administer Sun Cluster SNMP MIB *(Continued)*

Subcommand	Purpose
`show`	Displays configuration information for MIBs on the specified nodes.

TABLE A–17 `clsnmpuser`: Administer Sun Cluster SNMP Users

Subcommand	Purpose
`create`	Adds a user to the SNMP user configuration on the specified node.
`delete`	Deletes an SNMPv3 user from the specified node.
`export`	Exports the SNMP user information from the specified node.
`list`	Prints a list of SNMPv3 users that are configured on the specified node.
`set`	Sets the configuration of a user on the specified node.
`set-default`	Sets the default user and security level to use when sending traps using SNMPv3.
`show`	Displays the information about the users on the specified node.

TABLE A–18 `cltelemetryattribute`, `clta`: Configure System Resource Monitoring

Subcommand	Purpose
`disable`	Disables the specified telemetry attribute for the specified object type.
`enable`	Enables data collection for the specified telemetry attribute for the specified object types.
`export`	Exports the configuration of the telemetry attributes of object types and object instances to a file or to the standard output (`stdout`).
`list`	Displays the telemetry attributes that are configured for the specified object types.
`print`	Displays system resource usage for the specified telemetry attributes that are enabled for the specified object instances or object types.
`set-threshold`	Modifies the settings of a threshold for a specified telemetry attribute on a specified object on a node.
`show`	Displays the properties that are configured for telemetry attributes on object types or object instances.

TABLE A–19 `cluster:` Manage the Global Configuration and Status of a Cluster

Subcommand	Purpose
`check`	Checks and reports whether the cluster is configured correctly.

TABLE A–19 `cluster`: Manage the Global Configuration and Status of a Cluster *(Continued)*

Subcommand	Purpose
`create`	Creates a cluster by using configuration information that is stored in a `clconfigfile` file.
`export`	Exports the configuration information in a cluster configuration file.
`list`	Displays the name of the cluster on which you issue the cluster command.
`list-checks`	Prints a list with the check ID and description of each available check.
`list-cmds`	Prints a list of all available Sun Cluster commands.
`rename`	Renames the cluster on which you issue the cluster command.
`restore-netprops`	Repairs the cluster private-network settings of the cluster on which you issue the cluster command.
`set`	Sets the properties of the cluster on which you issue the cluster command.
`set-netprops`	Sets the properties of the cluster private network address.
`show`	Displays detailed configuration information about cluster components for the specified clusters.
`show-netprops`	Displays the private network address settings.
`shutdown`	Shuts down the cluster on which you issue the cluster command in an orderly fashion.
`status`	Displays the status of cluster components in the specified cluster.

TABLE A–20 `clvxvm`: Configure Veritas Volume Manager for Sun Cluster

Subcommand	Purpose
`encapsulate`	Encapsulates the root disk and performs other Sun Cluster-specific tasks.
`initialize`	Initializes VxVM and performs other Sun Cluster-specific tasks.

TABLE A–21 `clzonecluster`: Create and Manage Zone Clusters for Sun Cluster

Subcommand	Purpose
`boot`	Boots the zone cluster.
`clone`	Clones the zone cluster.
`configure`	Launches an interactive utility to configure and create a zone cluster.
`delete`	Removes a specific zone cluster.
`halt`	Stops a zone cluster or a specific node on the zone cluster.
`install`	Installs a zone cluster.

TABLE A–21 `clzonecluster`: Create and Manage Zone Clusters for Sun Cluster *(Continued)*

`list`	Displays the names of configured zone clusters.
`move`	Moves the zone path to a new zone path.
`ready`	Prepares the zone for applications.
`reboot`	Reboots a zone cluster.
`show`	Displays the properties of zone clusters.
`status`	Determines if the zone cluster node is a member of the zone cluster.
`uninstall`	Uninstalls a zone cluster.
`verify`	Checks that the syntax of the specified information is correct.

◆ ◆ ◆ APPENDIX B

Standard Properties

This appendix describes the standard resource type, resource, and resource group properties. This appendix also describes the resource property attributes that are available for changing system-defined properties and creating extension properties.

Note – Property names for resource types, resources, and resource groups are *not* case sensitive. You can use any combination of uppercase and lowercase letters when you specify property names.

This appendix covers the following topics:

Resource Type Properties

The following information describes the resource type properties that are defined by the Sun Cluster software.

The property values are categorized as follows:

- **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 space 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 space 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 or 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 space 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 was not declared in the RTR file.
- **Query-only** – Cannot be set directly by an administrative tool.

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.

Property names are shown first, followed by a description.

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 minimum version of the resource management API that is required to support this resource type implementation.

The following information summarizes the maximum `API_version` that is supported by each release of Sun Cluster.

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 2/08	8
3.2	9

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 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 that was released before 3.2.

Note – If you do not declare this property or set this property to the default value (2), the data service can be installed on any version of Sun Cluster starting with Sun Cluster 3.0.

Category: Optional

Default: `2`

Tunable: `NONE`

`Boot` (`string`)

An optional callback method that specifies the path to the `Boot` method program. The RGM runs the `Boot` method for each managed resource of this type, on a node that joins or rejoins the cluster.

The set of nodes on which `Boot`, `Init`, `Fini`, or `Validate` methods are run is determined by the setting of the resource types `Init_nodes` property. You can set the `Init_nodes` property to `RG_PRIMARIES`, which indicates the nodes that are specified in the resource type's `Installed_nodes` property.

Category: Conditional or Explicit

Default: No default

Tunable: `NONE`

`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 at the same time.

You use this resource-type property in combination with the `Scalable` resource property, as follows:

If the value of the `Failover` resource type is	If the value of the `Scalable` resource is	Description
`TRUE`	`TRUE`	Do not specify this illogical combination.
`TRUE`	`FALSE`	Specify this combination for a failover service.

If the value of the `Failover` resource type is	If the value of the `Scalable` resource is	Description
`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 to run in a global-cluster non-voting node. But, do not configure a scalable resource to run in multiple global-cluster non-voting nodes on the same Solaris host.
`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 of `Scalable` in the `r_properties(5)` man page 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:	`NONE`

`Fini` (`string`)

An optional callback method that specifies the path to the `Fini` method program. The RGM runs the `Fini` method when a resource of this type is no longer managed by the RGM.

The `Fini` method usually undoes any initializations that were performed by the `Init` method.

The set of nodes on which `Boot`, `Init`, `Fini`, or `Validate` methods are run is determined by the setting of the resource types `Init_nodes` property. You can set the `Init_nodes` property to `RG_PRIMARIES`, which indicates the nodes that are specified in the resource type's `Installed_nodes` property.

The RGM executes `Fini` on each node on which the resource becomes unmanaged when the following situations arise:

- The resource group that contains the resource is switched to an unmanaged state. In this case, the RGM executes the `Fini` method on all nodes in the node list.
- The resource is deleted from a managed resource group. In this case, the RGM executes the `Fini` method on all nodes in the node list.

- A node is deleted from the node list of the resource group that contains the resource. In this case, the RGM executes the `Fini` method on only the deleted node.

A “node list” is either the resource group's `Nodelist` or the resource type's `Installed_nodes` list. Whether “node list” refers to the resource group's `Nodelist` or the resource type's `Installed_nodes` list depends on the setting of the resource type's `Init_nodes` property. The `Init_nodes` property can be set to `RG_primaries` or `RT_installed_nodes`. For most resource types, `Init_nodes` is set to `RG_primaries`, the default. In this case, both the `Init` and `Fini` methods are executed on the nodes that are specified in the resource group's `Nodelist`.

The type of initialization that the `Init` method performs defines the type of cleanup that the `Fini` method that you implement needs to perform, as follows:

- Cleanup of node-specific configuration.
- Cleanup of cluster-wide configuration.

Category:	Conditional or Explicit
Default:	No default
Tunable:	`NONE`

`Global_zone` (`boolean`)
A Boolean value that, if declared in the RTR file, indicates whether the methods of this resource type execute in the global zone, that is, either a zone-cluster node or a global-cluster non-voting node. If this 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. 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.

Category:	Optional
Default:	`FALSE`
Tunable:	`ANYTIME`

`Init` (`string`)
An optional callback method that specifies the path to the `Init` method program. The RGM runs the `Init` method when a resource of this type becomes managed by the RGM.

The set of nodes on which `Boot`, `Init`, `Fini`, or `Validate` methods are run is determined by the setting of the resource types `Init_nodes` property. You can set the `Init_nodes` property to `RG_PRIMARIES`, which indicates the nodes that are specified in the resource type's `Installed_nodes` property.

Category:	Conditional or Explicit
Default:	No default
Tunable:	`NONE`

`Init_nodes` (enum)

Indicates the nodes 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 that can master the resource) or `RT_INSTALLED_NODES` (all nodes on which the resource type is installed).

Category:	Optional
Default:	`RG_PRIMARIES`
Tunable:	`NONE`

`Installed_nodes` (`string_array`)

A list of the cluster node names on which the resource type can be run. Specify an asterisk (*) to explicitly include all cluster nodes, which is the default.

Category:	The cluster administrator can configure this property
Default:	All cluster nodes
Tunable:	`ANYTIME`

`Is_logical_hostname` (boolean)

`TRUE` indicates that this resource type is some version of the `LogicalHostname` resource type that manages failover Internet Protocol (IP) addresses.

Category:	Query-only
Default:	No default
Tunable:	`NONE`

`Is_shared_address` (boolean)

`TRUE` indicates that this resource type is some version of the `SharedAddress` resource type that manages shared Internet Protocol (IP) addresses.

Category:	Query-only
Default:	No default
Tunable:	`NONE`

`Monitor_check` (`string`)

An optional callback method: the path to the program that the RGM runs before performing a monitor-requested failover of a resource of this type. If the monitor-check program exits with nonzero on a node, any attempt to fail over to that node as a result of calling `scha_control` with the `GIVEOVER` tag is prevented.

Category: Conditional or Explicit

Default: No default

Tunable: `NONE`

`Monitor_start` (`string`)

An optional callback method: the path to the program that the RGM runs to start a fault monitor for a resource of this type.

Category: Conditional or Explicit

Default: No default

Tunable: `NONE`

`Monitor_stop` (`string`)

A callback method that is required if `Monitor_start` is set: the path to the program that the RGM runs to stop a fault monitor for a resource of this type.

Category: Conditional or Explicit

Default: No default

Tunable: `NONE`

`Pkglist` (`string_array`)

An optional list of packages that are included in the resource type installation.

Category: Conditional or Explicit

Default: No default

Tunable: `NONE`

`Postnet_stop` (`string`)

An optional callback method: the path to the program that the RGM runs after calling the `Stop` method of any network-address resources on which a resource of this type depends. After the network interfaces are configured down, this method must perform `Stop` actions.

Category: Conditional or Explicit

Default: No default

Tunable: `NONE`

`Prenet_start` (`string`)

An optional callback method: the path to the program that the RGM runs before the RGM calls the `Start` method of any network-address resources on which a resource of this type depends. This method performs `Start` actions that must be performed before network interfaces are configured.

Category: Conditional or Explicit

Default: No default

Tunable: `NONE`

`Proxy` (`boolean`)

A Boolean value that 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. See the `scha_control`(1HA) and `scha_control`(3HA) man pages for more information.

If set 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 cluster administrator does not set this property directly. Rather, the RGM updates this property when the cluster administrator adds or removes a resource of this type to or from any resource group.

Category: Query-only

Default: Empty list

Tunable: `NONE`

`Resource_type` (`string`)

The name of the resource type. To view the names of the currently registered resource types, use:

```
resourcetype show +
```

In Sun Cluster 3.1 and Sun Cluster 3.2, a resource type name includes the version, which is mandatory:

vendor-id`.`*resource-type*`:`*rt-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 `resourcetype` 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, use the stock symbol of the company that is creating the resource type. Resource type names that were created before Sun Cluster 3.1 continue to use the syntax:

vendor-id`.`*resource-type*

Category: Required

Default: Empty string

Tunable: `NONE`

`RT_basedir` (`string`)
The directory path that is used to complete relative paths for callback methods. This path must be set to the directory in which the resource type packages are installed. The path must be a complete path, that is, it must start with a forward slash (`/`).

Category: Required unless all method path names are absolute

Default: No default

Tunable: `NONE`

`RT_description` (`string`)
A brief description of the resource type.

Category: Conditional

Default: Empty string

Tunable: `NONE`

`RT_system` (`boolean`)
If the `RT_system` property is `TRUE` for a resource type, you cannot delete the resource type (**`resourcetype unregister`** *resource-type-name*) . This property prevents the 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:	`ANYTIME`

`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. The `RT_version` is the suffix component of the full resource type name.

Category:	Conditional/Explicit or Required
Default:	No default
Tunable:	`NONE`

`Single_instance` (`boolean`)
If `TRUE`, indicates that only one resource of this type can exist in the cluster.

Category:	Optional
Default:	`FALSE`
Tunable:	`NONE`

`Start` (`string`)
A callback method: the path to the program that the RGM runs to start a resource of this type.

Category:	Required unless the RTR file declares a `Prenet_start` method
Default:	No default
Tunable:	`NONE`

`Stop` (`string`)
A callback method: the path to the program that the RGM runs to stop a resource of this type.

Category:	Required unless the RTR file declares a `Postnet_stop` method
Default:	No default
Tunable:	`NONE`

`Update` (`string`)
An optional callback method: the path to the program that the RGM runs when properties of a running resource of this type are changed.

Category:	Conditional or Explicit
Default:	No default
Tunable:	`NONE`

`Validate` (`string`)

An optional callback method that specifies the path to the `Validate` method program. The RGM runs the `Validate` method to check values for properties of resources of this type.

The set of nodes on which `Boot`, `Init`, `Fini`, or `Validate` methods are run is determined by the setting of the resource types `Init_nodes` property. You can set the `Init_nodes` property to `RG_PRIMARIES`, which indicates the nodes that are specified in the resource type's `Installed_nodes` property.

Category:	Conditional or Explicit
Default:	No default
Tunable:	`NONE`

`Vendor_ID` (`string`)

See the `Resource_type` property.

Category:	Conditional
Default:	No default
Tunable:	`NONE`

Resource Properties

This section describes the resource properties that are defined by the Sun Cluster software.

The property values are categorized as follows:

- **Required**. The cluster administrator must specify a value when he or she creates a resource with an administrative utility.
- **Optional**. If the cluster administrator does not specify a value when he or she creates a resource group, the system supplies a default value.
- **Conditional**. The RGM creates the property only if the property is declared in the RTR file. Otherwise, the property does not exist and is not available to cluster administrators. 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 `Tunable` attribute, which is described in “Resource Property Attributes” on page 219, lists whether and when you can update resource properties, as follows:

`FALSE` or `NONE`	Never
`TRUE` or `ANYTIME`	Any time

`AT_CREATION`	When the resource is added to a cluster
`WHEN_DISABLED`	When the resource is disabled

Property names are shown first, followed by a description.

`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.

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`.

This property is used only for scalable services.

Category:	Optional
Default:	No default
Tunable:	`ANYTIME`

`Boot_timeout` for each callback method in the Type (`integer`)

A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category:	Conditional or Optional
Default:	`3600` (one hour), if the method itself is declared in the RTR file
Tunable:	`ANYTIME`

`Cheap_probe_interval` (`integer`)

The number of seconds between invocations of a quick fault probe of the resource. This property is created by the RGM and is 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 RTR file, the `Tunable` value for the property is `WHEN_DISABLED`.

Category:	Conditional
Default:	No default
Tunable:	`WHEN_DISABLED`

Extension properties

Extension properties as declared in the RTR file of the resource's type. The implementation of the resource type defines these properties. "Resource Property Attributes" on page 219 contains information about the individual attributes that you can set for extension properties.

Category: Conditional

Default: No default

Tunable: Depends on the specific property

Failover_mode (enum)

Modifies the recovery actions that the RGM takes when a resource fails to start or to 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. See the scha_control(1HA) and the scha_control(3HA) man pages. NONE indicates that the RGM is not to take any recovery action when one of the previously listed 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. 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 to force the resource group offline. The RGM might then attempt to start the group on another node.

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 (scha_control). 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 node is neither failed over nor 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 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 and 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 number of 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.

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` and 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` and 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 the `scha_control()` function 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 either 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` and 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 either to restart the resource or resource group or to fail over the group fail.

Category:	Optional
Default:	`NONE`
Tunable:	`ANYTIME`

`Fini_timeout` for each callback method in the Type (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category:	Conditional or Optional
Default:	`3600` (one hour), if the method itself is declared in the RTR file

Tunable: `ANYTIME`

`Global_zone_override` (`boolean`)

This property is allowed only for resource types that set the `Global_zone=TRUE` property in the RTR file. The setting of the `Global_zone_override` property overrides the value of the resource type property `Global_zone` for the particular resource. See the `rt_properties(5)` man page for more information.

When the `Global_zone` property is set to `TRUE` the resource methods always execute in the global-cluster voting node.

Setting the `Global_zone_override` property to `FALSE` forces the resource methods to execute on a non-global zone, that is, either a zone-cluster node or a global-cluster non-voting node in which the resource group is configured, rather than always executing in the global zone as they usually would when the `Global_zone` property is set to `TRUE`.

This property is optional if a default value is specified in the RTR file.

If the `Tunable` attribute is not specified in the RTR file, the `Tunable` value for the property is `AT_CREATION`. You can set the `Tunable` attribute in the RTR file to `AT_CREATION`, `WHEN_DISABLED`, or `ANYTIME`.

Use caution when you set the `Tunable` attribute to `Anytime` in the RTR file. Changes to the `Global_zone_override` property take effect immediately, even if the resource is online. For example, suppose that the `Global_zone_override` tunability is set to `ANYTIME` and the `Global_zone_override` property is currently set to `FALSE` on a resource that is configured in a non-global zone. When the resource is switched online, the starting methods are executed in the non-global zone. If the `Global_zone_override` property is then set to `TRUE` and the resource is switched offline, the stopping methods are executed in the global zone. Your method code must be able to handle this possibility. If it cannot, then you must set the `Tunable` attribute to `WHEN_DISABLED` or `AT_CREATION` instead.

Category: Conditional or Optional

Default: `TRUE`

Tunable: `AT_CREATION`

`Init_timeout` for each callback method in the Type (`integer`)

A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category: Conditional or Optional

Default: `3600` (one hour), if the method itself is declared in the RTR file

Tunable: `ANYTIME`

`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`. A given client (identified by the client IP address) of the scalable service is always sent to the same node of the cluster.

`Lb_sticky_wild`. A given 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 the IP address is coming.

Category: Conditional or 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 that is distributed to the specified *node*. The fraction of load that is 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 one-fourth of the load and node 2 receives three-fourths of the load. The empty string (“”), the default, sets a uniform distribution. Any node that is not assigned an explicit weight receives a default weight of 1.

If the `Tunable` attribute is not specified in the RTR file, the `Tunable` value for the property is `ANYTIME`. Changing this property revises the distribution for new connections only.

Category: Conditional or Optional

Default: The empty string (“”)

Tunable: `ANYTIME`

`Monitor_check_timeout` for each callback method in the Type (`integer`)

A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category: Conditional or Optional

Default: `3600` (one hour), if the method itself is declared in the RTR file

Tunable: `ANYTIME`

`Monitor_start_timeout` for each callback method in the Type (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category:	Conditional or Optional
Default:	`3600` (one hour), if the method itself is declared in the RTR file
Tunable:	`ANYTIME`

`Monitor_stop_timeout` for each callback method in the Type (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category:	Conditional or Optional
Default:	`3600` (one hour), if the method itself is declared in the RTR file
Tunable:	`ANYTIME`

`Monitored_switch` (`enum`)
Set to `Enabled` or `Disabled` by the RGM if the cluster administrator enables or disables the monitor with an administrative utility. If `Disabled`, monitoring on the resource is stopped, although the resource itself remains online. The `Monitor_start` method is not called until monitoring is re-enabled. If the resource does not have a monitor callback method, this property does not exist.

Category:	Query-only
Default:	No default
Tunable:	`NONE`

`Network_resources_used` (`string_array`)
A list of logical-hostname or shared-address network resources on which the 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 `Scalable` is not declared in the RTR file, `Network_resources_used` is unavailable unless it is explicitly declared in the RTR file.

This property is updated automatically by the RGM, based on the setting of the resource-dependencies properties. You do not need to set this property directly. However, if you add a resource name to this property, the resource name is automatically added to the `Resource_dependencies` property. In addition, if you delete a resource name from this property, the resource name is automatically deleted from any resource-dependencies property in which the resource also appears.

Category:	Conditional or Optional
Default:	The empty list
Tunable:	`ANYTIME`

`Num_resource_restarts` on each cluster node (`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`(1HA) command with the `RESOURCE_RESTART` argument.
- The `scha_control`(3HA) 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 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 (`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`(1HA) command with the `RESTART` argument.
- The `scha_control`(3HA) 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`)
Set to `Enabled` or `Disabled` by the RGM if the cluster administrator enables or disables the resource with an administrative utility. If disabled, a resource is brought offline and has no callbacks run until it is re-enabled.

Category:	Query-only
Default:	No default
Tunable:	`NONE`

`Port_list` (`string_array`)
A 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`.

You can specify the following protocol values:

- `tcp`, for TCP IPv4
- `tcp6`, for TCP IPv6
- `udp`, for UDP IPv4
- `udp6`, for 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 Apache is described in the *Sun Cluster Data Service for Apache Guide for Solaris OS*.

Category:	Conditional or Required
Default:	No default
Tunable:	`ANYTIME`

`Postnet_stop_timeout` for each callback method in the Type (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category:	Conditional or Optional
Default:	`3600` (one hour), if the method itself is declared in the RTR file
Tunable:	`ANYTIME`

`Prenet_start_timeout` for each callback method in the Type (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category:	Conditional or Optional
Default:	`3600` (one hour), if the method itself is declared in the RTR file
Tunable:	`ANYTIME`

`R_description` (`string`)
A brief description of the resource.

Category:	Optional
Default:	The empty string
Tunable:	`ANYTIME`

`Resource_dependencies` (`string_array`)
A list of resources on which the resource has a strong dependency. A strong dependency determines the order of method calls.

A resource with resource dependencies, referred to as the *dependent resource*, cannot be started if any resource in the list, referred to as the *depended-on resource*, is not online. If the dependent resource and one of the depended-on resources in the list start at the same time, the RGM waits to start the dependent resource until the depended-on resource in the list starts. If the depended-on resource does not start, the dependent resource remains offline. The depended-on resource might not start because the resource group for the depended-on resource in the list remains offline or is in a `Start_failed` state. If the dependent resource remains offline because of a dependency on a depended-on resource in a different resource group that fails to start or is disabled or offline, the dependent resource's group enters a `Pending_online_blocked` state. If the dependent resource has a dependency on a depended-on resource in the same resource group that fails to start or is disabled or offline, the resource group does not enter a `Pending_online_blocked` state.

By default in a resource group, application resources have an implicit strong resource dependency on network address resources. `Implicit_network_dependencies` in "Resource Group Properties" on page 205 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-host basis. The behavior of the dependent is affected by the depended-on resource only on the same host. The dependent resource waits for the depended-on resource to start on the same host. 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: `ANYTIME`

`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, and the resource cannot recover, the RGM brings that resource on that node 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 re-enables 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-host basis. The behavior of the dependent is affected by the depended-on resource only on the same host. The dependent resource waits for the depended-on resource to start on the same host. 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: `ANYTIME`

`Resource_dependencies_restart` (`string_array`)

A list of resources on which the resource has a restart dependency. A restart dependency determines the order of method calls.

This property works as `Resource_dependencies` does, with one addition. If any resource in the restart dependency list, referred to as a *depended-on resource*, is restarted, the resource with resource dependencies, referred to as the *dependent resource*, is restarted. After the depended-on resource in the list comes back online, the RGM stops and restarts the dependent resource. This restart behavior occurs when the resource groups that contain the dependent and depended-on resources remain online.

A resource with resource dependencies, referred to as the *dependent resource*, cannot be started if any resource in the list, referred to as the *depended-on resource*, is not online. If the dependent resource and one of the depended-on resources in the list start at the same time, the RGM waits to start the dependent resource until the depended-on resource in the list starts. If the depended-on resource does not start, the dependent resource remains offline. The depended-on resource might not start because the resource group for the depended-on resource in the list remains offline or is in a `Start_failed` state. If the dependent resource remains offline because of a dependency on a depended-on resource in a different resource group that fails to start or is disabled or offline, the dependent resource's group enters a `Pending_online_blocked` state. If the dependent resource has a dependency on a depended-on resource in the same resource group that fails to start or is disabled or offline, the resource group does not enter a `Pending_online_blocked` state.

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-host basis. The behavior of the dependent is affected by the depended-on resource only on the same host. The dependent resource waits for the depended-on resource to start on the same host. 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:	`ANYTIME`

`Resource_dependencies_weak` (`string_array`)

A list of resources on which the resource has a weak dependency. A weak dependency determines the order of method calls.

The RGM calls the `Start` methods of the resources in this list, referred to as the *depended-on resources*, before the `Start` method of the resource with resource dependencies, referred to as the *dependent resource*. The RGM calls the `Stop` methods of the dependent resource before the `Stop` methods of the depended-on resources. The dependent resource can still start if the depended-on resources fail to start or remain offline.

If the dependent resource and a depended-on resource in its `Resource_dependencies_weak` list start concurrently, the RGM waits to start the dependent resource until the depended-on resource in the list starts. If the depended-on resource in the list does not start, for example, if the resource group for the depended-on resource in the list remains offline or the depended-on resource in the list is in a `Start_failed` state, the dependent resource starts. The dependent resource's resource group might enter a `Pending_online_blocked` state temporarily as resources in the dependent resource's `Resource_dependencies_weak` list start. When all depended-on resources in the list have started or failed to start, the dependent resource starts and its group reenters the `Pending_online` state.

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-host basis. The behavior of the dependent is affected by the depended-on resource only on the same host. The dependent resource waits for the depended-on resource to start on the same host. 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: `ANYTIME`

`Resource_name` (`string`)

The name of the resource instance. This name must be unique within the cluster configuration and cannot be changed after a resource has been created.

Category: Required

Default: No default

Tunable: `NONE`

`Resource_project_name` (`string`)

The Solaris project name that is 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 starts the related processes under this project name. If this property is not specified, the project name is 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 OS.

Note – Changes to this property take effect the next time that the resource is started.

Category: Optional

Default: Null

Tunable: `ANYTIME`

`Resource_state` on each cluster node (`enum`)

The RGM-determined state of the resource on each cluster node. Possible states are `Online`, `Offline`, `Start_failed`, `Stop_failed`, `Monitor_failed`, `Online_not_monitored`, `Starting`, and `Stopping`.

You cannot configure this property.

Category:	Query-only
Default:	No default
Tunable:	`NONE`

`Retry_count` (`integer`)

The number of times that 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 perform one of the following actions:

- Allow the resource group to remain on the current primary node, even though the resource is in a faulted state
- Request a failover of the resource group onto a different node

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 RTR file, the `Tunable` value for the property is `WHEN_DISABLED`.

Note – If you specify a negative value for this property, the monitor attempts to restart the resource an unlimited number of times.

However, 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 over 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 is 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 RTR file, the `Tunable` value for the property is `WHEN_DISABLED`.

Category: Conditional

Default: No 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 the Sun Cluster software.

Note – You can configure a scalable resource group (which uses network load-balancing) to run in a global-cluster non-voting node. However, you can run such a scalable resource group in only one node per Solaris host.

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 cannot 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. These properties can be useful in a nonscalable service as well as in a scalable service.

Using this resource property in combination with the `Failover` resource type property is described in more detail in the `r_properties(5)` man page.

Category: Optional

Default: No default

Tunable: `AT_CREATION`

`Start_timeout` for each callback method in the Type (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category:	Conditional or Optional
Default:	`3600` (one hour), if the method itself is declared in the RTR file
Tunable:	`ANYTIME`

`Status` on each cluster node (`enum`)
Set by the resource monitor with the `scha_resource_setstatus` command or the `scha_resource_setstatus()` or `scha_resource_setstatus_zone()` functions. Possible values are `OK`, `DEGRADED`, `FAULTED`, `UNKNOWN`, and `OFFLINE`. When a resource is brought online or offline, the RGM automatically sets the `Status` value if the `Status` value is not set by the resource's monitor or methods.

Category:	Query-only
Default:	No default
Tunable:	`NONE`

`Status_msg` on each cluster node (`string`)
Set by the resource monitor at the same time as the `Status` property. When a resource is brought online or offline, the RGM automatically resets this property to the empty string if this property is not set by the resource's methods.

Category:	Query-only
Default:	No default
Tunable:	`NONE`

`Stop_timeout` for each callback method in the Type (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category:	Conditional or Optional
Default:	`3600` (one hour), if the method itself is declared in the RTR file
Tunable:	`ANYTIME`

`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 is 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 RTR file, the `Tunable` value for the property is `WHEN_DISABLED`.

Category:	Conditional
Default:	No default

Tunable: `WHEN_DISABLED`

`Type` (`string`)
The resource type of which this resource is an instance.

Category: Required

Default: No default

Tunable: `NONE`

`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 might appear as a suffix of the resource type name. When a resource is edited, the `Type_version` property can be changed to a new value.

The tunability of this property is derived from the following sources:

- The current version of the resource type
- The `#$upgrade_from` directive in the RTR file

Category: See description

Default: No default

Tunable: See description

`UDP_affinity` (`boolean`)
If this property is set to `TRUE`, sends all UDP traffic from a given client 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`.

This property is only used for scalable services.

Category: Optional

Default: No default

Tunable: `WHEN_DISABLED`

`Update_timeout` for each callback method in the Type (`integer`)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category: Conditional or Optional

Default: `3600` (one hour), if the method itself is declared in the RTR file

Tunable: ANYTIME

Validate_timeout for each callback method in the Type (integer)
A time lapse, in seconds, after which the RGM concludes that an invocation of this method has failed. For a given resource type, timeout properties are defined only for those methods that are declared in the RTR file.

Category: Conditional or Optional

Default: 3600 (one hour), if the method itself is declared in the RTR file

Tunable: ANYTIME

Weak_affinity (boolean)
If this property is set to TRUE, this property enables the weak form of the client affinity.

The weak form of the client affinity allows connections from a given client to be sent to the same server node except when the following conditions occur:

- A server listener starts in response to, for example, a fault monitor's restarting, a resource's failing over or switching over, or a node's rejoining a cluster after failing
- Load_balancing_weights for the scalable resource changes because the cluster administrator performed an administrative 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: Optional

Default: No default

Tunable: WHEN_DISABLED

Resource Group Properties

The following information describes the resource group properties that are defined by the Sun Cluster software.

The property values are categorized as follows:

- **Required.** The cluster administrator must specify a value when creating a resource group 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.

- **Query-only.** Cannot be set directly by an administrative tool.

Property names are shown first, followed by a description.

`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 of 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 online` command or the equivalent GUI instruction. After that, the resource group resumes normal failover behavior.

Category:	Optional
Default:	`TRUE`
Tunable:	`ANYTIME`

`Desired_primaries` (integer)
The preferred number of nodes that the group can run on 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.

Category:	Optional
Default:	`1`
Tunable:	`ANYTIME`

`Failback` (boolean)
A Boolean value that indicates whether to recalculate the set of nodes on which the group is online when a node joins the cluster. A recalculation can cause the RGM to bring the group offline on less preferred nodes and online on more preferred nodes.

Category:	Optional
Default:	`FALSE`
Tunable:	`ANYTIME`

`Global_resources_used` (string_array)
Indicates whether cluster file systems are used by any resource in this resource group. Legal values that the cluster administrator can specify are an asterisk (*) to indicate all global resources, and the empty string (“”) to indicate no global resources.

Category:	Optional
Default:	All global resources

Tunable: ANYTIME

`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.

Category: Optional

Default: `TRUE`

Tunable: ANYTIME

`Maximum_primaries` (integer)

The maximum number of nodes on which the 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.

Category: Optional

Default: `1`

Tunable: ANYTIME

`Nodelist` (string_array)

A list of cluster nodes on which a resource group can be brought online in order of preference. These nodes are known as the potential primaries or masters of the resource group.

Category: Optional

Default: The list of all voting nodes in the cluster in arbitrary order

Tunable: ANYTIME

`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.

Category: Optional

Default: The empty string

Tunable: ANYTIME

`Pingpong_interval` (`integer`)

A nonnegative integer value (in seconds) that is used by the RGM to determine where to bring the resource group online in these instances:

- In the event of a reconfiguration.
- As the result of the execution of a `scha_control` command with the `GIVEOVER` argument or the `scha_control()` function with the `SCHA_GIVEOVER` argument.

In the event of a reconfiguration, the resource group might fail more than once to come online within the past `Pingpong_interval` seconds on a particular node. This failure occurs because the resource's `Start` or `Prenet_start` method exited with a nonzero status or timed out. As a result, that node is considered ineligible to host the resource group, and the RGM looks for another master.

If a `scha_control` command or `scha_control -O GIVEOVER` command is executed on a given node by a resource, thereby causing its resource group to fail over to another node, the first node (on which `scha_control` was run) cannot be the destination of another `scha_control -O GIVEOVER` by the same resource until `Pingpong_interval` seconds have elapsed.

Category:	Optional
Default:	`3600` (one hour)
Tunable:	`ANYTIME`

`Resource_list` (`string_array`)

The list of resources that are contained in the group. The cluster administrator does not set this property directly. Rather, the RGM updates this property as the cluster administrator adds or removes resources from the resource group.

Category:	Query-only
Default:	No default
Tunable:	`NONE`

`RG_affinities` (`string`)

The RGM is to try to locate a resource group on a host that is a current master of another given resource group (positive affinity) or that is not a current master of a given resource group (negative affinity).

You can set `RG_affinities` to the following strings:

- ++, or strong positive affinity
- +, or weak positive affinity
- -, or weak negative affinity
- --, or strong negative affinity
- +++, or strong positive affinity with failover delegation

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 the `RG_affinities` property is described in Chapter 2, "Administering Data Service Resources."

Category: Optional

Default: The empty string

Tunable: `ANYTIME`

`RG_dependencies` (`string_array`)
Optional list of resource groups that indicates a preferred ordering for bringing other groups online or offline on the same node. 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, that is, RG1 has a resource group dependency on RG2.

The following list summarizes the effects of this resource group dependency:

- When a node joins the cluster, `Boot` methods on that node are not run on resources in RG1 until all `Boot` methods on that node have completed on resources in RG2.
- If RG1 and RG2 are both in the `PENDING_ONLINE` state on the same node at the same time, the starting methods (`Prenet_start` or `Start`) are not run on any resources in RG1 until all the resources in RG2 have completed their starting methods.
- If RG1 and RG2 are both in the `PENDING_OFFLINE` state on the same node at the same time, the stopping methods (`Stop` or `Postnet_stop`) are not run on any resources in RG2 until all the resources in RG1 have completed their stopping methods.
- An attempt to switch the primaries of RG1 or RG2 fails if switching the primaries would leave RG1 online on any node and RG2 offline on all nodes. The `clresourcegroup`(1CL) and `clsetup`(1CL) man pages contain more information.
- 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.

Category: Optional

Default: The empty list

Tunable: `ANYTIME`

`RG_description` (`string`)
A brief description of the resource group.

Category: Optional

Default:	The empty string
Tunable:	`ANYTIME`

`RG_is_frozen` (`boolean`)
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.

Category:	Optional
Default:	No default
Tunable:	`NONE`

`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.

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 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`.

Category:	Optional
Default:	Depends on the value of `Maximum_primaries`
Tunable:	`NONE`

`RG_name` (`string`)
The name of the resource group. This property is required and must be unique within the cluster.

Category:	Required
Default:	No default
Tunable:	`NONE`

`RG_project_name` (`string`)
The Solaris project name (see the `projects(1)` man page) 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

starts the related processes under this project name for resources that do not have the `Resource_project_name` property set (see the `r_properties(5)` man page). 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 OS.

Note – Changes to this property take affect the next time that the resource is started.

Category: Optional

Default: The text string "`default`"

Tunable: `ANYTIME`

`RG_slm_cpu` (`decimal number`)

If the `RG_slm_type` property is set to `AUTOMATED`, this number is the basis for the calculation of the number of CPU shares and the size of the processor set.

Note – You can only use the `RG_slm_cpu` property if `RG_slm_type` is set to `AUTOMATED`. For more information, see the `RG_slm_type` property.

The maximum value for the `RG_slm_cpu` property is 655. You can include two digits after the decimal point. Do not specify 0 for the `RG_slm_cpu` property. If you set a share value to 0, a resource might not be scheduled by the Fair Share Scheduler (FFS) when the CPU is heavily loaded.

Changes that you make to the `RG_slm_cpu` property while the resource group is online are taken into account dynamically.

Because the `RG_slm_type` property is set to `AUTOMATED`, Sun Cluster creates a project named `SCSLM_`*resourcegroupname*. *resourcegroupname* represents the actual name that you assign to the resource group. Each method of a resource that belongs to the resource group is executed in this project. Starting with Solaris 10 OS, these projects are created in the resource group's node, whether it is a global-cluster voting node or a global-cluster non-voting node. See the `project(4)` man page.

The project `SCSLM_`*resourcegroupname* has a `project.cpu-shares` value of 100 times the `RG_slm_cpu` property value. If the `RG_slm_cpu` property is not set, this project is created with a `project.cpu-shares` value of 1. The default value for the `RG_slm_cpu` property is `0.01`.

Starting with the Solaris 10 OS, if the `RG_slm_pset_type` property is set to `DEDICATED_STRONG` or to `DEDICATED_WEAK`, the `RG_slm_cpu` property is used to calculate the size of processor sets. The `RG_slm_cpu` property is also used to calculate the value of `zone.cpu-shares`.

For information about processor sets, see the *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

Note – You can use the `RG_slm_cpu` property only on a global cluster. You cannot use this property on a zone cluster.

Category:	Optional
Default:	`0.01`
Tunable:	`ANYTIME`

`RG_slm_cpu_min` (`decimal number`)
Determines the minimum number of processors on which an application can run.

You can only use this property if all of the following factors are true:

- The `RG_slm_type` property is set to `AUTOMATED`
- The `RG_slm_pset_type` property is set to `DEDICATED_STRONG` or to `DEDICATED_WEAK`
- The `RG_slm_cpu` property is set to a value that is greater than or equal to the value set for the `RG_slm_cpu_min` property
- You are using the Solaris 10 OS

The maximum value for the `RG_slm_cpu_min` property is 655. You can include two digits after the decimal point. Do not specify 0 for the `RG_slm_cpu_min` property. The `RG_slm_cpu_min` and `RG_slm_cpu` properties determine the values of `pset.min` and `pset.max`, respectively, for the processor set that Sun Cluster generates.

Changes that you make to the `RG_slm_cpu` and the `RG_slm_cpu_min` properties while the resource group is online are taken into account dynamically. If the `RG_slm_pset_type` property is set to `DEDICATED_STRONG`, and not enough CPUs are available, the change that you request for the `RG_slm_cpu_min` property is ignored. In this case, a warning message is generated. On next switchover, if not enough CPUs are available for the `RG_slm_cpu_min` property, errors due to the lack of CPUs can occur.

For information about processor sets, see the *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

Note – You can use the `RG_slm_cpu_min` property only on a global cluster. You cannot use this property on a zone cluster.

Category:	Optional
Default:	`0.01`

Tunable: `ANYTIME`

`RG_slm_type` (`string`)

Enables you to control system resource usage and automate some steps to configure the Solaris operating system for system resource management. Possible values for `RG_SLM_type` are `AUTOMATED` and `MANUAL`.

If you set the `RG_slm_type` property to `AUTOMATED`, the resource group is started with control of the CPU usage.

As a result, Sun Cluster does the following:

- Creates a project named `SCSLM_`*resourcegroupname*. 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.
- Sets the value of `project.cpu_shares` that is associated with the project to the value of the `RG_slm_cpu` property times 100. By default, the value for `project.cpu_shares` is 1.
- Starting with the Solaris 10 OS, sets `zone.cpu_shares` to 100 times the sum of the `RG_slm_cpu` property in all the online resource groups. This property also sets `RG_slm_type` to `AUTOMATED` in this node. The node can be a global-cluster voting node or a global-cluster non-voting node. The global-cluster non-voting node is bound to a Sun Cluster generated pool. Optionally, if the `RG_slm_pset_type` property is set to `DEDICATED_WEAK` or to `DEDICATED_STRONG`, this Sun Cluster generated pool is associated with a Sun Cluster generated processor set. For information about dedicated processor sets, see the description of the `RG_slm_pset_type` property. When you set the `RG_slm_type` property to `AUTOMATED`, all operations that are performed are logged.

If you set the `RG_slm_type` property to `MANUAL`, the resource group executes in the project that is specified by the `RG_project_name` property.

For information about resource pools and processor sets, see the *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`.

Note – You can use the RG_slm_type property only on a global cluster. You cannot use this property on a zone cluster.

Category:	Optional
Default:	manual
Tunable:	ANYTIME

RG_slm_pset_type (string)
Enables the creation of a dedicated processor set.

You can only use this property if all of the following factors are true:

- The RG_slm_type property is set to AUTOMATED
- You are using the Solaris 10 OS
- The resource group executes in a global-cluster non-voting node

Possible values for RG_slm_pset_type are DEFAULT, DEDICATED_STRONG, and DEDICATED_WEAK.

For a resource group to execute as DEDICATED_STRONG or DEDICATED_WEAK, the resource group must be configured so there are only global-cluster non-voting nodes in its node list.

The global-cluster non-voting node must not be configured for a pool other than the default pool (POOL_DEFAULT). For information about zone configuration, see the zonecfg(1M) man page. This global-cluster non-voting node must not be dynamically bound to a pool other than the default pool. For more information about pool binding, see the poolbind(1M) man page. These two pool conditions are verified only when the methods of the resources in the resource group are launched.

The values DEDICATED_STRONG and DEDICATED_WEAK are mutually exclusive for resource groups that have the same node in their node list. You cannot configure resource groups in the same node so that some have RG_slm_pset_type set to DEDICATED_STRONG and others set to DEDICATED_WEAK.

If you set the RG_slm_pset_type property to DEDICATED_STRONG, Sun Cluster does the following in addition to the actions performed by the RG_slm_type property when it is set to AUTOMATED:

- Creates and dynamically binds a pool to the global-cluster non-voting node in which the resource group starts for either or both the PRENET_START and START methods.
- Creates a processor set with a size between the following sums
  - The sum of the RG_slm_cpu_min property in all the resource groups that are online on the node in which this resource group starts.
  - The sum of the RG_slm_cpu property in the resource groups that are running on that node.

When either the `STOP` or `POSTNET_STOP` methods execute, the Sun Cluster generated processor set is destroyed. If resource groups are no longer online on the node, the pool is destroyed, and the global-cluster non-voting node is bound to the default pool (`POOL_DEFAULT`).

- Associates the processor set to the pool.
- Sets `zone.cpu_shares` to 100 times the sum of the `RG_slm_cpu` property in all the resource groups that are running the node.

If you set the `RG_slm_pset_type` property to `DEDICATED_WEAK`, the resource group behaves the same as if `RG_slm_pset_type` was set to `DEDICATED_STRONG`. However, if enough processors are not available to create the processor set, the pool is associated to the default processor set.

If you set the `RG_slm_pset_type`property to `DEDICATED_STRONG` and not enough processors are available to create the processor set, an error is generated. As a result, the resource group is not started on that node.

When CPUs are allocated, the `DEFAULTPSETMIN` minimum size has priority over `DEDICATED_STRONG`, which has priority over `DEDICATED_WEAK`. However, when you use the `clnode` command to increase the size of the default processor, and not enough processors are available, this priority is ignored. For information about the `DEFAULTPSETMIN` property, see the `clnode`(1CL) man page.

The `clnode` command assigns a minimum of CPUs to the default processor set dynamically. If the number of CPUs that you specify is not available, Sun Cluster periodically retries to assign this number of CPUs. Failing that, Sun Cluster tries to assign smaller numbers of CPUs to the default processor set until the minimum number of CPUs are assigned. This action might destroy some `DEDICATED_WEAK` processor sets, but does not destroy `DEDICATED_STRONG` processor sets.

When you start a resource group for which you've set the `RG_slm_pset_type` property to `DEDICATED_STRONG`, it might destroy the processor sets that are associated with the `DEDICATED_WEAK` processor sets. This resource group might do so if not enough CPUs are available on the node for both processor sets. In this case, the processes of the resource group that are running in the `DEDICATED_WEAK` processor sets are associated with the default processor set.

To swap the value of the `RG_slm_pset_type` property between `DEDICATED_STRONG` or `DEDICATED_WEAK`, you must first set it to the default.

If resource groups that are configured for CPU control are not online in a global-cluster non-voting node, the CPU share value is set to `zone.cpu-shares` for that node. By default, `zone.cpu-shares` is set to 1. For more information about zone configuration, see the `zonecfg`(1M) man page.

If you set the `RG_slm_pset_type` property to `DEFAULT`, Sun Cluster creates a pool named `SCSLM_pool_`*zonename*, but does not create a processor set. In this case, `SCSLM_pool_`*zonename* is associated with the default processor set. The shares that are assigned to the node equal the sum of the values for `RG_slm_cpu` for all the resource groups in the node.

For information about resource pools and processor sets, see the *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

Note – You can use the `RG_slm_pset_type` property only on a global cluster. You cannot use this property on a zone cluster.

Category:	Optional
Default:	`default`
Tunable:	`ANYTIME`

`RG_state` on each cluster node (`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 group on each cluster node.

You cannot configure this property. However, you can indirectly set this property by running the `clresourcegroup` command or by using the equivalent `clsetup` or Sun Cluster Manager commands. A 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 only, except the `UNMANAGED` state, which applies across all nodes. For example, a resource group might be `OFFLINE` on node A, but `PENDING_ONLINE` on node B.

`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.
`ONLINE`	The resource group has been started on the node. In other words, the starting methods

	Prenet_start, Start, and Monitor_start, as applicable to each resource, have executed successfully on all enabled resources in the group.
OFFLINE	The resource group has been stopped on the node. In other words, the stopping 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.
PENDING_ONLINE	The resource group is starting on the node. The starting methods Prenet_start, Start, and Monitor_start, as applicable to each resource, are being executed on enabled resources in the group.
PENDING_OFFLINE	The resource group is stopping on the node. The stopping methods Monitor_stop, Stop, and Postnet_stop, as applicable to each resource, are being executed on enabled resources in the group.
ERROR_STOP_FAILED	One or more resources within the resource group failed to stop successfully and are in the Stop_failed state. Other resources in the group might remain online or offline. This resource group is not permitted to start on any node until the ERROR_STOP_FAILED state is cleared. You must use an administrative command, such as clresource clear, to manually kill the Stop_failed resource and reset its state to OFFLINE.
ONLINE_FAULTED	The resource group was PENDING_ONLINE and has finished starting on this node. However, one or more resources ended up in the START_FAILED state or with FAULTED status.
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.

Category: Query-only

Default: No default

Tunable: NONE

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. You can also still enable and 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.

You do 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.

Category: Query-only

Default: FALSE

Tunable: NONE

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 clresourcegroupcommand is 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.

Operation	Example
Delete a resource group	**`clresourcegroup delete RG1`**
Edit a resource group property (except for `RG_system`)	**`clresourcegroup set -p RG_desription=... +`**
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 in a resource group	**`clresource enable R1`**
Enable monitoring for a resource in a resource group	**`clresource monitor R1`**
Disable a resource in a resource group	**`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.

Category: Optional

Default: `FALSE`

Tunable: `ANYTIME`

Resource Property Attributes

This section describes the resource property attributes that you can use to change system-defined properties or to create extension properties.

Caution – You cannot specify `Null` or the empty string (“”) as the default value for `boolean`, `enum`, or `int` types.

Property names are shown first, followed by a description.

`Array_maxsize`
For a `stringarray` type, the maximum number of array elements that are permitted.

`Array_minsize`
For a `stringarray` type, the minimum number of array elements that are permitted.

`Default`
Indicates a default value for the property.

`Description`
A string annotation that is 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 that are permitted for the property.

`Extension`
If used, indicates that the RTR file entry declares an extension property that is defined by the resource type implementation. Otherwise, the entry is a system-defined property.

`Max`
For an `int` type, the maximum value that is permitted for the property.

`Maxlength`
For `string` and `stringarray` types, the maximum string length that is permitted.

`Min`
For an `int` type, the minimal value that is permitted for the property.

`Minlength`
For `string` and `stringarray` types, the minimum string length that is permitted.

`Per_node`
If used, indicates that the extension property can be set on a per-node 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 property value on a node to which an explicit value has not been assigned.

You cannot specify the `Per_node` property attribute for a property of type `stringarray`.

`Property`
The name of the resource property.

`Tunable`
Indicates when the cluster administrator can set the value of this property in a resource. Set to `NONE` or `FALSE` to prevent the cluster administrator from setting the property. Values that enable a cluster administrator to tune a property are `TRUE` or `ANYTIME` (at any time),

AT_CREATION (only when the resource is created), or WHEN_DISABLED (when the resource is disabled). To establish other conditions, such as "when monitoring is disabled" or "when offline", set this attribute to ANYTIME and validate the state of the resource in the Validate method.

The default differs for each standard resource property, as shown in the following entry. The default setting for tuning an extension property, if not otherwise specified in the RTR file, is TRUE (ANYTIME).

Type of the property

Allowable types are string, boolean, integer, 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.

APPENDIX C

Legal RGM Names and Values

This appendix lists the requirements for legal characters for Resource Group Manager (RGM) names and values.

This appendix covers the following topics:

- "RGM Legal Names" on page 223
- "RGM Values" on page 225

RGM Legal Names

RGM names fall into the following categories:

- Resource group names
- Resource type names
- Resource names
- Property names
- Enumeration literal names

Rules for Names Except Resource Type Names

Except for resource type names, all names must comply with these rules:

- Names must be in ASCII.
- Names must start with a letter.
- Names can contain uppercase and lowercase letters, digits, dashes (-), and underscores (_).
- The maximum number of characters that you can use in a name is 255.

Format of Resource Type Names

The format of the complete name of a resource type depends on the resource type, as follows:

- If the resource type's resource type registration (RTR) file contains the `#$upgrade` directive, the format is as follows:

  vendor-id.*base-rt-name*:*rt-version*
- If the resource type's RTR file does *not* contain the `#$upgrade` directive, the format is as follows:

  vendor-id.*base-rt-name*

A period separates *vendor-id* and *base-rt-name*. A colon separates *base-rt-name* and *rt-version*.

The variable elements in this format are as follows:

vendor-id	Specifies the vendor ID prefix, which is the value of the `Vendor_id` resource type property in the RTR file. If you are developing a resource type, choose a vendor ID prefix that uniquely identifies the vendor, such as your company's stock ticker symbol. For example, the vendor ID prefix of resource types that are developed by Sun Microsystems, Inc. is `SUNW`.
base-rt-name	Specifies the base resource type name, which is the value of the `Resource_type` resource type property in the RTR file.
rt-version	Specifies the version suffix, which is the value of the `RT_version` resource type property in the RTR file. The version suffix is *only* part of the complete resource type name if the RTR file contains the `#$upgrade` directive. The `#$upgrade` directive was introduced in Release 3.1 of the Sun Cluster product.

Note – If only one version of a base resource type name is registered, you do not have to use the complete name in administrative commands. You can omit the vendor ID prefix, the version number suffix, or both.

For more information, see "Resource Type Properties" on page 175.

EXAMPLE C–1 Complete Name of a Resource Type With the `#$upgrade` Directive

This example shows the complete name of a resource type for which properties in the RTR file are set, as follows:

- `Vendor_id=SUNW`
- `Resource_type=sample`
- `RT_version=2.0`

The complete name of the resource type that is defined by this RTR file is as follows:

EXAMPLE C–1 Complete Name of a Resource Type With the `#$upgrade` Directive *(Continued)*

```
SUNW.sample:2.0
```

EXAMPLE C–2 Complete Name of a Resource Type Without the `#$upgrade` Directive

This example shows the complete name of a resource type for which properties in the RTR file are set, as follows:

- `Vendor_id=SUNW`
- `Resource_type=nfs`

The complete name of the resource type that is defined by this RTR file is as follows:

```
SUNW.nfs
```

RGM Values

RGM values fall into two categories: property values and description values. Both categories share the same rules:

- Values must be in ASCII.
- The maximum length of a value is 4 megabytes minus 1, that is, 4,194,303 bytes.
- Values cannot contain the following characters:
  - Null
  - Newline
  - Comma (,)
  - Semicolon (;)

APPENDIX D

Data Service Configuration Worksheets and Examples

This appendix provides worksheets for planning resource-related components of your cluster configuration and examples of completed worksheets for your reference. For worksheets for other components of your cluster configuration. see Appendix A, "Sun Cluster Installation and Configuration Worksheets," in *Sun Cluster Software Installation Guide for Solaris OS*.

If necessary, make additional copies of a worksheet to accommodate all the resource-related components in your cluster configuration. To complete these worksheets, follow the planning guidelines in *Sun Cluster Software Installation Guide for Solaris OS* and Chapter 1, "Planning for Sun Cluster Data Services." Then refer to your completed worksheets during cluster installation and configuration.

Note – The data that is used in the worksheet examples is intended as a guide only. The examples do not represent a complete configuration of a functional cluster.

Configuration Worksheets

This appendix contains the following worksheets.

Resource Types Worksheet

Use this worksheet for resource types other than logical host or shared address.

Resource Type Name	Nodes on Which the Resource Type Runs

EXAMPLE D–1 Resource Types Worksheet

Resource Type Name	Nodes on Which the Resource Type Runs
`SUNW.nshttp`	`phys-schost-1, phys-schost-2`
`SUNW.oracle_listener`	`phys-schost-1, phys-schost-2`
`SUNW.oracle_server`	`phys-schost-1, phys-schost-2`

Network Resources Worksheet

Component	Name	
Resource name		
Resource group name		
Resource type (circle one)	Logical hostname | Shared address	
Resource type name		
Dependencies		
Hostnames used		
Extension properties	**Name**	**Value**

EXAMPLE D–2 Network Resources—Shared Address Worksheet

Component	Name	
Resource name	**sh-galileo**	
Resource group name	**rg-shared**	
Resource type (circle one)	**Shared address**	
Resource type name	**SUNW.SharedAddress**	
Dependencies	**none**	
Hostnames used	**sh-galileo**	
Extension properties	**Name**	**Value**
	netiflist	**ipmp0@1, ipmp0@2**

EXAMPLE D–3 Network Resources—Logical Hostname Worksheet

Component	Name	
Resource name	**relo-galileo**	
Resource group name	**rg-oracle**	
Resource type (circle one)	**Logical hostname**	
Resource type name	**SUNW.LogicalHostname**	
Dependencies	**none**	
Hostnames used	**relo-galileo**	
Extension properties	**Name**	**Value**
	netiflist	**ipmp0@1, ipmp0@2**

Application Resources—Failover Worksheet

Component	Name	
Resource name		
Resource group name		
Resource type name		
Dependencies		
Extension Properties	**Name**	**Value**

EXAMPLE D–4 Application Resources—Failover Worksheet

Component	Name	
Resource name	`oracle-listener`	
Resource group name	`rg-oracle`	
Resource type name	`SUNW.oracle_listener`	
Dependencies	`hasp_resource`	
Extension Properties	**Name**	**Value**
	`ORACLE_HOME`	`/global/oracle/orahome/`
	`LISTENER_NAME`	`lsnr1`

Application Resources—Scalable Worksheet

Component	Name	
Resource name		
Logical-host resource group name		
Shared-address resource group name		
Logical-host resource type name		
Shared-address resource type name		
Dependencies		
Extension Properties	**Name**	**Value**

EXAMPLE D–5 Application Resources—Scalable Worksheet

Component	Name	
Resource name	`sh-galileo`	
Logical-host resource group name		
Shared-address resource group name	`rg-shared`	
Logical-host resource type name		
Shared-address resource type name		
Dependencies		
Extension Properties	**Name**	**Value**

Resource Groups—Failover Worksheet

Component	Notes	Name
Resource group name	The name must be unique within the cluster	
Function	Describe the function of this resource group	
Failback? (circle one)	Will this resource group switch back to the primary node after the primary node has failed and been restored?	Yes | No
Node list	Indicate the cluster nodes that can host this resource group. The first node in this list should be the primary, with others being the secondaries. The order of the secondaries will indicate preference for becoming primaries.	
Dependent disk device groups	List the disk device groups on which this resource group depends.	
Configuration directory	If the resources in this resource group need to create files for administrative purposes, include the subdirectory they use.	

EXAMPLE D–6 Example: Resource Groups—Failover Worksheet

Component	Notes	Name
Resource group name	The name must be unique within the cluster	`rg-oracle`
Function	Describe the function of this resource group	`Contains the Oracle resources`
Failback? (circle one)	Will this resource group switch back to the primary node after the primary node has failed and been restored?	`No`

Node list	Indicate the cluster nodes that can host this resource group. The first node in this list should be the primary, with others being the secondaries. The order of the secondaries will indicate preference for becoming primaries.	`1) phys-schost-1`
		`2) phys-schost-2`

Dependent disk device groups	List the disk device groups on which this resource group depends.	`schost1-dg`
Configuration directory	If the resources in this resource group need to create files for administrative purposes, include the subdirectory they use.	

Resource Groups—Scalable Worksheet

Component	Notes	Name
Resource group name	The name must be unique within the cluster.	
Function		
Maximum number of primaries		
Desired number of primaries		
Failback? (circle one)	Will this resource group switch back to the primary node, after the primary node has failed?	Yes | No
Node list	Indicate the cluster nodes that can host this resource group. The first node in this list should be the primary, with others being the secondaries. The order of the secondaries will indicate preference for becoming primaries.	
Dependencies	List any resource groups on which this resource depends.	

EXAMPLE D–7 Example: Resource Groups—Scalable Worksheet

Component	Notes	Name
Resource group name	The name must be unique within the cluster.	`rg-http`
Function		`Contains the web server resources`
Maximum number of primaries		`2`
Desired number of primaries		`2`
Failback? (circle one)	Will this resource group switch back to the primary node, after the primary node has failed?	`No`

Node list	Indicate the cluster nodes that can host this resource group. The first node in this list should be the primary, with others being the secondaries. The order of the secondaries will indicate preference for becoming primaries.	`1) phys-schost-1`
		`2) phys-schost-2`

Dependencies	List any resource groups on which this resource depends.	`rg-shared`

Index