

VERITAS Cluster Server™ 1.3.0

User's Guide

UNIX and Windows NT

October 2000
30-000034-399


VERITAS

Disclaimer

The information contained in this publication is subject to change without notice. VERITAS Software Corporation makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. VERITAS Software Corporation shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual.

Copyright

Copyright © 1998-2000 VERITAS Software Corporation. All rights reserved. VERITAS is a registered trademark of VERITAS Software Corporation in the US and other countries. The VERITAS logo and VERITAS Cluster Server are trademarks of VERITAS Software Corporation. All other trademarks or registered trademarks are the property of their respective owners.

Printed in the USA, October 2000.

VERITAS Software Corporation
1600 Plymouth St.
Mountain View, CA 94043
Phone 650-335-8000
Fax 650-335-8050
www.veritas.com



Contents

Preface	xi
How This Guide Is Organized	xi
Technical Support	xii
For Customers Within the U.S. and Canada	xii
For Customers Outside the U.S. and Canada	xii
Conventions	xiii
Chapter 1. Introducing VCS	1
What is a VCS Cluster?	2
Core VCS Processes	3
VCS Communications: GAB and LLT	4
How GAB Operates	4
Managing Cluster Memberships	4
Monitoring Heartbeats	4
Distributing Information	5
How LLT Operates	5
VCS Components	6
Resources	6
Resource Types	6
Service Groups	6
Attributes	7
How VCS Controls and Monitors Resources	7
Agents	7
Resource Dependencies	8



Establishing Resource Dependencies According to Type	8
Additional Considerations for Resource Type Dependencies	9
Bringing Resources Online	10
The VCS Violation Event Trigger	11
Chapter 2. The VCS Environment	13
Shared Storage	13
Communication Channels	14
Multiple Communication Channels	14
Configuration Options for Membership Services (GAB)	15
Service Group Heartbeat Disks (UNIX Only)	15
Disk Regions	15
Chapter 3. The VCS Configuration Language	17
How the Configuration File Defines a Cluster	18
About VCS Attributes	18
Include Clause	20
Resource Type Definitions	20
About the VCS Name Rule	21
Cluster Definitions	23
System Definitions	23
Domain-Qualified System Names–UNIX Only	24
SNMP Definitions	25
Service Group Definitions	26
Service Group Attributes	26
Resource Definitions	26
Resource Attributes	27
Resource Dependencies	27



Managing the VCS Configuration File	28
The hacf Utility	28
Verifying a Configuration	28
Generating a main.cmd file from main.cf	28
Generating a main.cf file from main.cmd	28
Loading a Configuration	29
Dumping a Running Configuration	29
Multiple Versions of .cf Files	29
Sample Configuration Files	30
Sample main.cf File for UNIX	30
Sample types.cf File for UNIX	33
Sample main.cf File for Windows NT	37
Sample types.cf File for Windows NT	39
Chapter 4. Administering VCS From the Command Line	45
VCS Environment Variables	45
How VCS Identifies the Local System	47
Starting VCS	47
Starting VCS on UNIX	48
Starting VCS on Windows NT	48
Stopping VCS	49
For UNIX Users	49
Additional Considerations for Stopping VCS	50
Adding, Modifying, and Deleting Users	50
Guidelines	50
Configuration Read/Write and Read-Only Modes	50
Querying VCS	52
Resources	52
Service Groups	54
Systems	55



Resource Types	55
Resource Type Agents	55
Clusters	56
Status Queries	57
Basic Operations	58
System Operations	58
Forcing a System to Start	58
Modifying System Attributes	58
Changing a System's Load Attribute	58
Displaying System Nodeid Value	58
Freezing a System	58
Thawing a System	59
Service Group Operations	59
Onlining a Group	59
Offlining a Group	59
Switching a Group	59
Freezing a Group	59
Thawing a Group	59
Enabling a Group	60
Disabling a Group	60
Enabling Group Resources	60
Disabling Group Resources	60
Flushing a Group	61
Resource Operations	61
Onlining a Resource	61
Offlining a Resource	61
Probing a Resource	62
Clearing a Resource	62
Agent Operations	62



Reconfiguring VCS	63
Configuration Read/Write and Read-Only Modes	63
The -modify Syntax	63
Localizing Attributes	63
Adding, Modifying, and Deleting Resource Types	66
Adding, Modifying, and Deleting Attributes	67
Adding Service Groups	68
Constraints	69
Modifying Service Group System Lists	70
Constraints	70
Adding Resources	72
Constraints	72
Linking Resources	73
Constraints	73
Deleting and Unlinking Service Groups, and Resources	74
Constraints	74
VCS High-Level Configuration Language and Low-Level Commands	75
Sample Configuration on UNIX	75
Chapter 5. About the VCS GUI	77
Before Using Cluster Manager	77
Setting the Display	78
About Cluster Manager Users	78
Adding A User Account from the Command Prompt	79
Starting Cluster Manager	80
Adding and Configuring a Cluster Panel	81
The Cluster Manager Windows	82
The Cluster Monitor	83
Cluster Explorer	84
The Cluster Explorer Toolbar	85



Cluster Explorer View Panels	87
The Command Center	92
Supported Commands	93
The Cluster Shell	95
The Template View Window	96
About VCS Templates	97
Managing the Cluster	98
Logging On to and Off of a Cluster	99
Managing Users from Cluster Manager	99
Setting User Preferences	103
Font/Color Display	103
Sound Information	104
Getting Status Information on the Cluster and Its Objects	105
Opening, Saving, and Closing a Cluster Configuration	105
Monitoring Cluster Objects	106
Monitoring Object Attributes	106
Monitoring Resource Dependencies	107
Monitoring Service Group Dependencies	108
Monitoring Heartbeats	108
Monitoring Resource Types	108
Adding and Deleting Service Groups	109
Adding and Deleting Resources	111
Adding and Deleting Systems	113
Managing Resource and Service Group Dependencies	114
Managing Systems for a Service Group	118
Bringing Service Groups Online	119
Taking Service Groups Offline	120
Bringing Resources Online	121
Taking Resources Offline	122
Creating a New Service Group Using the Wizard	123



Viewing Log Information	125
Importing Additional Resource Types Information	126
Configuration Editor	127
Chapter 6. Service Group Dependencies	129
What is a Service Group Dependency?	129
Why Configure a Service Group Dependency?	130
Categories of Service Group Dependencies	131
Online Group Dependency	131
Offline Group Dependency	131
Location of Dependency	132
Type of Dependency	134
Soft Dependency	134
Firm Dependency	135
Overview of Service Group Dependency Configurations	136
Failover Parent/Failover Child	136
Failover Parent/Parallel Child	138
Parallel Parent/Failover Child	140
Parallel Parent/Parallel Child	142
Configuring Service Group Dependencies	144
Dependency Limitations	144
Automatic Actions for Service Group Dependencies	145
Automatic Online	145
Automatic Failover	145
Manual Operations for Service Group Dependencies	146
Manual Online	146
Manual Offline	147
Manual Switch	147
Linking Service Groups (Online/Offline Group Dependencies)	148
Constraints	148



Chapter 7. VCS Performance Considerations	149
Impact of VCS on Overall System Performance	149
Kernel Components (GAB and LLT)	149
VCS Engine	150
Agents	151
Resource Type and Agent Configuration	151
Additional Considerations for Agents	152
VCS Performance	153
Cluster Boot Time	153
Bringing a Resource Online	153
Taking a Resource Offline	154
Bringing a Service Group Online	154
Taking a Service Group Offline	154
Detecting Resource Failure	154
Detecting System Failure	155
Detecting Network Link Failure	156
Service Group Switch	156
Service Group Failover	156
 Chapter 8. Advanced Topics	 157
Event Notification	157
How VCS Performs Event Notification	157
Sample Scripts	158
Types of Event Triggers	158
InJeopardy Event Trigger	159
NoFailover Event Trigger	159
PostOffline Event Trigger	160
PostOnline Event Trigger	160
PreOnline Event Trigger	161
ResFault Event Trigger	162



ResNotOff Event Trigger	163
SysOffline Event Trigger	164
Violation Event Trigger	164
Scheduling Class and Priority Configuration Support	165
Additional Information for Windows NT Users	165
Priority Ranges	165
Default Scheduling Classes and Priorities	166
VCS Security	166
Issuing Commands from the Command-Line Interface and Cluster Shell	166
Controlling Connections to VCS	166
Handling Network Failure	167
About Cluster Memberships	167
Disabling Failover	169
Example of How VCS Handles Network Failure	170
Appendix A. Troubleshooting VCS	179
Starting VCS on Windows NT	179
Startup Errors	180
Network Partitioning	182
Reconnecting the Private Network	183
Network Partitions and the UNIX Boot Monitor	183
Preexisting Network Partitions	183
VCS Seeding	184
Bringing Service Groups Online	184
Bringing Resources Online	187
Taking Service Groups Offline	188
Taking Resources Offline	189
Stopping VCS Without -force Option	189
Connecting to Cluster Manager from the Command Line	190
Primary Domain Controller File	190



SNMP Traps	191
Sample Trap Messages	191
When VCS Shuts Down a System	192
Appendix B. System States	193
Examples of State Transitions	195
Appendix C. VCS Attributes	197
Resource Attributes	197
Type-Specific Resource Attributes	201
Additional Attributes for Scheduling Class and Priority Configuration Support	203
Initializing Attributes in the Configuration File	203
Setting Attributes Dynamically from the Command Line	204
Service Group Attributes	205
System Attributes	212
Resource Type Attributes	216
Cluster Attributes	220
Additional Attributes for Scheduling Class and Priority Configuration Support	222
Initializing Attributes in the Configuration File	222
Setting Attributes Dynamically from the Command Line	223
SNMP Attributes	224
Index	225



Preface

How This Guide Is Organized

Chapter 1, “Introducing VCS,” explains important VCS concepts, including the relationship between service groups, resources, and attributes, and how a cluster operates. This chapter also introduces the core VCS processes.

Chapter 2, “The VCS Environment,” presents various topics on the VCS configuration, including shared storage, communication channels, and how VCS guards against network partitioning.

Chapter 3, “The VCS Configuration Language,” describes the VCS configuration language, including definitions, clauses, and dependencies.

Chapter 4, “Administering VCS From the Command Line,” describes how to start and stop VCS from the command line, how to monitor and administer service groups, and how to add, modify, and delete users from the cluster. This chapter also provides low-level commands to reconfigure various objects in the cluster and describes the relationship between the commands and the configuration language.

Chapter 5, “About the VCS GUI,” provides an overview of the VCS graphical user interface and configuration tool, including the service group configuration wizard.

Chapter 6, “Service Group Dependencies,” Defines the concept of a service group dependency, and explains their default behavior. This chapter also explains the various categories, locations, and types of dependencies.

Chapter 7, “VCS Performance Considerations,” describes the impact of VCS on overall system performance, and includes methods on how to adjust VCS to meet specific configuration requirements.



[Chapter 8, “Advanced Topics,”](#) contains miscellaneous advanced topics, such as event notification, the role of “jeopardy” in handling network failure, and VCS security.

[Appendix A, “Troubleshooting VCS,”](#) provides additional information on basic VCS operations on the UNIX platform, including starting and stopping service groups and resources. Future releases will include troubleshooting tips for Windows NT.

[Appendix B, “System States,”](#) describes the various system states and the order in which a system transitions from one state to another.

[Appendix C, “VCS Attributes,”](#) lists the VCS attributes for each cluster object, including service groups, resources, resource types, systems, clusters, and SNMP.

Technical Support

For assistance with this product, or information regarding VERITAS service packages, contact Technical Support at the numbers listed below.

For Customers Within the U.S. and Canada

VCS on UNIX

- ◆ Phone: 800.342.0652
- ◆ Email: support@veritas.com

VCS on Windows NT

- ◆ Phone: 800.634.4747. To access Fast Code, enter 100 after the phone number.
- ◆ Email: helpdesk@support.veritas.com

For Customers Outside the U.S. and Canada

From Europe, the Middle East, or Asia, visit the Technical Support website at <http://support.veritas.com> for a list of each country’s contact information.



Conventions

Typeface	Usage
<code>courier</code>	computer output, command references within text
<code>courier</code> (bold)	user input and commands, keywords in grammar syntax
<i>italic</i>	new terms, book titles, emphasis
<i>italic</i>	variables within a command
Symbol	Usage
#	UNIX superuser prompt (for all shells)
C:\>	DOS command prompt





Welcome to VERITAS Cluster Server™ (VCS), the latest high-availability solution for cluster configurations. VCS enables you to monitor systems and application services, and to restart services on a different system when hardware or software fails. Its enhanced feature set offers:

- ◆ Increased scalability.

VCS provides a framework that supports parallel applications as they distribute load on multiple systems. It monitors the application services, and can be configured to fail over faulted instances to alternate systems.

- ◆ Enhanced shared storage.

VCS supports shared storage on clusters of up to 32 systems connected by any combination of systems, disks, and other devices. VCS is designed to grow with your organization, enabling you to migrate from shared SCSI clusters to fibre-channel and other SAN technologies.

- ◆ Simplified administration.

VCS's replication technology provides a single-system view of administration. Systems share identical copies of configuration and resource states, enabling you to monitor and control applications and reconfigure the cluster from any system. Using the Java-based graphical user interface, you can administer multiple clusters from any UNIX or Windows NT workstation. VCS even supports online addition of new systems: you can add systems without taking down applications or the VCS services that monitor them.

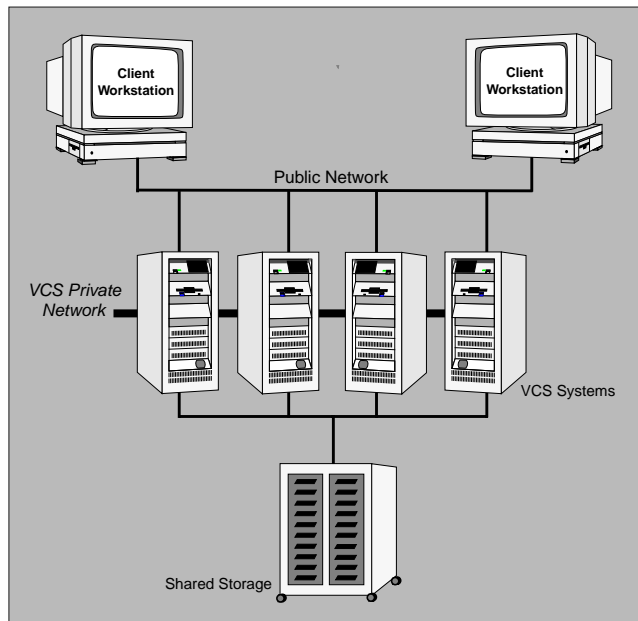


What is a VCS Cluster?

A single VCS cluster consists of multiple systems connected in various combinations to shared storage devices. VCS monitors and controls applications running in the cluster, and restarts applications in response to a variety of hardware or software faults. Client applications continue operation with little or no downtime. In some cases, such as virtual fileshare, this continuation is transparent to the user and high-level applications. In other cases, the operation must be retried; for example, reloading a Web page.

The VCS engine runs on each system in the cluster, and each system is a replicated state machine. VCS monitors the systems and their services over a private network. The systems communicate via *heartbeats* over an additional private network. This enables them to recognize which systems are active members of the cluster, which are joining or leaving the cluster, and which have failed. Client workstations receive service over the public network from applications running on the VCS systems.

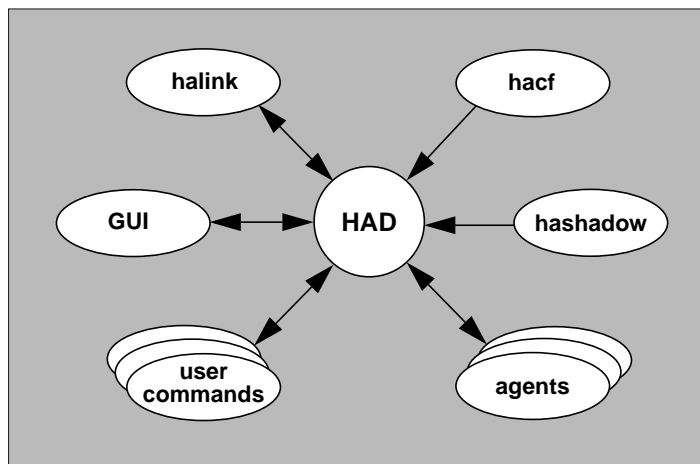
The following illustration shows a typical VCS configuration of four systems connected to shared storage.



Example of a Four-System VCS Cluster

Core VCS Processes

The core VCS processes shown in the figure below run on each system in the cluster.



Process	Definition
HAD	The VCS engine that maintains configuration information, administers policy making, and directs failover.
<i>Clockwise from right:</i>	
hacf	A utility that is executed on demand. It is the translator between the configuration language and the VCS engine.
hashadow	A process that monitors and restarts the VCS engine.
agents	There is one agent for each resource type. An agent brings online, takes offline, and monitors resources of its specific type.
user commands	A comprehensive set of commands to view and manipulate applications and objects.
GUI	A Java-based graphical user interface for viewing and manipulating applications. The VCS GUI, called "Cluster Manager," can run on any system inside or outside the cluster.
halink	A process that monitors communication links between systems in the cluster. The engine takes this data into account when deciding where to fail over a service group. This process is not started at default.



VCS Communications: GAB and LLT

Communications within a VCS environment are conducted by the Group Atomic Broadcast mechanism (GAB) and the Low Latency Transport mechanism (LLT). These kernel components are used only by VCS, and replace the functions of TCP/IP for VCS private network communications.

How GAB Operates

GAB performs three major functions:

- ◆ Manages cluster memberships.
- ◆ Monitors heartbeat communication on disk and Ethernet.
- ◆ Distributes information throughout the cluster.

Managing Cluster Memberships

Because GAB is a global mechanism, all systems within the cluster are immediately notified of changes in resource status, cluster membership, and configuration. GAB is also atomic, meaning that it continuously maintains a synchronized state in the cluster membership and configuration files of all cluster systems. If a failure occurs while transmitting status changes, GAB's atomicity ensures that, upon recovery, all systems will have the same information regarding the status of any monitored resource in the cluster.

Monitoring Heartbeats

GAB also monitors heartbeat communication between systems. Heartbeats are signals that are sent periodically from one system to another to verify that the systems are active. You may manually configure the heartbeat interval and specify the number of consecutive heartbeats that a system can miss before it determines that another system has failed.

When a system suspects that another system has failed, the system in question is probed by other systems in the cluster to verify the failure. If the system remains unresponsive, it is marked **DOWN** and excluded from the cluster. Its applications are then migrated to the other systems. GAB ensures that when this process begins, all remaining systems in the cluster have the same information regarding the status of the failed system and the migration of the applications. Note that GAB may kill the VCS engine when the engine is unresponsive or when systems previously disconnected are reconnected.

Distributing Information

GAB distributes information to all systems throughout the cluster regarding system loads, agent reports, and administrative commands. GAB can also be configured to track and distribute additional information. Visit the VERITAS Customer Support website (support.veritas.com) for a list of commands available for GAB.

How LLT Operates

LLT provides kernel-to-kernel communications and monitors network communications. LLT can be configured to:

- ◆ Set system IDs within a cluster.
- ◆ Set cluster IDs for multiple clusters.
- ◆ Tune network parameters such as heartbeat frequency.

LLT runs directly on top of the Data Link Protocol Interface (DLPI) layer on UNIX, and the Network Driver Interface Specification (NDIS) on Windows NT. This ensures that events such as state changes are reflected more quickly, which in turn enables faster responses. You may also configure LLT to run as “low priority.” This prevents VCS communication on the public network until the public network is the final link, thereby reducing the rate of heartbeat broadcasts. See the *VERITAS Installation Guide* for instructions on how to configure LLT.



VCS Components

In addition to the systems and processes that form a VCS cluster, there are other components integral to cluster design, including resources, resource types, service groups, and attributes.

Resources

Resources are hardware or software entities, such as disks, network interface cards (NICs), IP addresses, applications, and databases, which work together to provide a service to clients in a client/server environment. They are identified by unique names within the cluster. With VCS, you can view available resources from the graphical user interface “Cluster Manager,” or from the command line.

Typically, resources are dependent upon each other or related resources, meaning that they operate in a particular order as established in the configuration file. This is most important during *failover*, when resources must be brought online or taken offline in the correct order so they can migrate to another system in the cluster. Resource dependencies are described on page 8.

Resources monitored by VCS, but not brought online or taken offline, are referred to as *persistent resources*. VCS also supports resources that can be brought online and monitored, but cannot be taken offline.

Resource Types

Resources are classified according to *types*. Multiple resources can be of a single type; for example, two disk resources within a service group are both classified as type Disk. VCS includes a set of predefined resources types. You can also define your own and create resources of that type.

Service Groups

Service groups are composed of related resources. When a service group is brought online, all the resources within the group are brought online. There are two types of service groups, *failover* and *parallel*.

- ◆ A failover group is a service group that can be fully or partially online on only one system at a time. Most application services, such as virtual fileshare, are configured as failover groups.
- ◆ A parallel group is a service group that can be fully or partially online on more than one system at a time. Parallel application services, such as Oracle Parallel Server (OPS), are configured as parallel groups.

Attributes

Attributes contain data regarding the cluster, systems, service groups, resources, resource types, and agents. For example, the value of a service group's SystemList attribute specifies on which systems the group is configured, and the priority of each system within the group. Attributes are defined in the VCS configuration language. See "[About VCS Attributes](#)" on page 18 for more information.

For example the State attribute for a resource specifies the current state of the resource. Note that resources of a particular type contain the same attributes; however, their values are assigned per resource. Predefined attributes are associated with all VCS resources. You define additional attributes when you define resource types.

How VCS Controls and Monitors Resources

Controlling a resource means bringing it online (starting) and taking it offline (stopping). How VCS starts and stops a resource is specific to the resource type. For example, a file system resource is started by mounting it, and an IP resource is started by configuring the IP address on a network interface card.

Monitoring a resource means determining if it is online or offline. How VCS monitors a resource is also specific to the resource type. For example, a file system resource reports as online if mounted, and an IP address tests as online if configured.

Agents

Agents are VCS processes that bring resources online and take them offline. They also monitor resources and report any state changes to VCS.

VCS includes agents for predefined resource types, and an environment for developing new agents for the resource types you define. See the accompanying installation guide for information on agents included with VCS. See the *VERITAS Cluster Server Agent Developer's Guide* for information on developing custom agents.



Resource Dependencies

Dependencies between resources specify the order in which the resources within a service group are brought online and taken offline. Resources and their dependencies form a *graph*. The resources at the top of the graph are *root* resources. Resources at the bottom of the graph are *leaf* resources. Parent resources appear at the top of the arcs that connect them to their child resources. Typically, child resources are brought online before parent resources, and parent resources are taken offline before child resources. Resources must adhere to the established order of dependency.

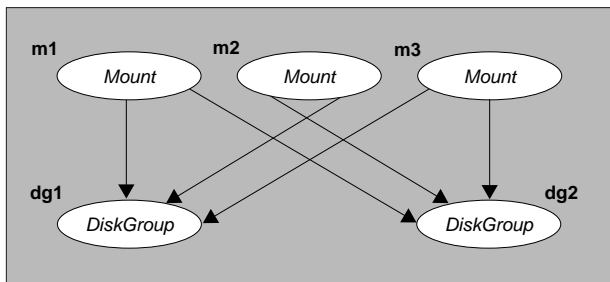
Establishing Resource Dependencies According to Type

VCS provides a method for establishing dependencies between resources according to the resource type. Resource-type dependencies are specified by the user on a per-group basis in the main.cf file. Rather than specifying the resource dependencies one-by-one, you can specify that all resources of a particular type depend on all resources of another type within the service group.

For example, to configure all resources of type `Mount` to depend on all resources of type `DiskGroup`, define the `TypeDependencies` service group attribute as:

```
TypeDependencies = { Mount, DiskGroup }
```

The VCS engine then generates links from each `Mount` resource to each `DiskGroup` resource within the service group, as illustrated below. For more information on attributes, see [Appendix C](#).



Note that existing resource-to-resource dependencies remain valid. Like resource dependencies, type-based dependencies do not allow cyclical relationships, nor can you specify resource-to-resource dependencies that form cyclical relationships with type-based dependencies.

For example:

```
TypeDependencies = { Mount, DiskGroup }
```

Here, resource `m1` is of type `Mount`, and it depends on resource `dg1` of type `DiskGroup`. You cannot configure the following resource dependency:

```
dg1 requires m1
```

Additional Considerations for Resource Type Dependencies

Persistent resource types (types for which the `Operations` attribute is set to `None` or `OnOnly`) typically appear as “leaves” in the dependency graph, meaning that they do not depend on other resources. If they appear elsewhere in the graph, resources listed prior to the persistent resource depend on the persistent resource and on the next, non-persistent type listed in the attribute.

For example:

```
TypeDependencies = { T1, T2, T3 }
```

In this example, if `T2` is a persistent type, all resources of type `T1` will depend on all resources of `T2` and `T3`.

Note It is essential that all Persistent resource types used within the `TypeDependencies` attribute are used in the configuration. For example, if `TypeDependencies = { T1, T2, T8, T3, T4 }`, and if type `T8` were not in the configuration, a disjointed dependency graph would be created. It is likely that resources of type `T2` would fault while attempting to be brought online before resources of type `T3` were online.

You can modify the `TypeDependencies` attribute using the `hagrpd -modify` command. Note that the following commands may affect a configuration using type-based dependencies.

- ◆ `hares -add resource resource_type service_group`

In this example, if the `resource_type` is listed in the `TypeDependencies` attribute for the `service_group`, the `resource` will be linked automatically, according to the type-based rule.

- ◆ `hares -delete resource`

In this example, if the `resource_type` is listed in the `TypeDependencies` attribute for the `service_group`, the `resource` will be unlinked, and the result will be the same as deleting a resource from a conventional dependency (similar to `requires` or `hares -link`).



Bringing Resources Online

A service group's resources are brought online according to how the group was started, as described below. For more information on the conditions required for a service group to be brought online, see "[Why Configure a Service Group Dependency?](#)" on page 130.

The service group was brought online by a command.

In this case, resources whose attributes are set to start automatically (the default setting) are brought online. However, before they can start, all of their child resources must first be brought online.

The service group was brought online due to a system booting.

If a system is a member of a failover service group's AutoStartList attribute (described on page 205), and if it is not already running on another system in the cluster, the group is brought online when the system is booted. Resources and their dependencies set to start automatically are brought online after their child resources.

If the system is a member of a parallel service group's AutostartList attribute, the group is brought online when the system is booted.

The group was brought online due to a failover.

VCS attempts to bring resources online that were already online on the failed system, or were in the process of going online. As in the previous examples, each parent resource must wait for its child resources to be brought online before starting.

The VCS Violation Event Trigger

The VCS violation event trigger notifies the administrator when a failover service group is online on more than one system (concurrency violation). This can occur if a group is fully or partially online on one system, and you bring the service group fully or partially online on another system outside of VCS control; that is, without using `hagrp -online` or `hares -online`. For example, a service group that includes the Oracle resource is online on `sysa`, and you start the Oracle resource using the Oracle command on `sysb`.

This trigger is invoked only on the system that caused the concurrency violation, in the above example, `sysb`. Specifically, it takes the service group offline on the system where the trigger was invoked.

This trigger is installed in the following directories by default:

- ◆ On **UNIX**: `$VCS_HOME/bin/triggers/violation`
- ◆ On **Windows NT**: `VCS_HOME\bin\triggers\Violation`

Note that if the file is not in the default location cited above, copy and modify the template file from the following directories:

- ◆ On **UNIX**: `$VCS_HOME/bin/sample_triggers/violation`
- ◆ On **Windows NT**: `VCS_HOME\bin\sample_triggers\Violation`

The violation event trigger can send a message to the system log and console on all cluster systems, and may be customized to send additional messages or emails. For information on this and other event triggers, see “[Event Notification](#)” on page 157.

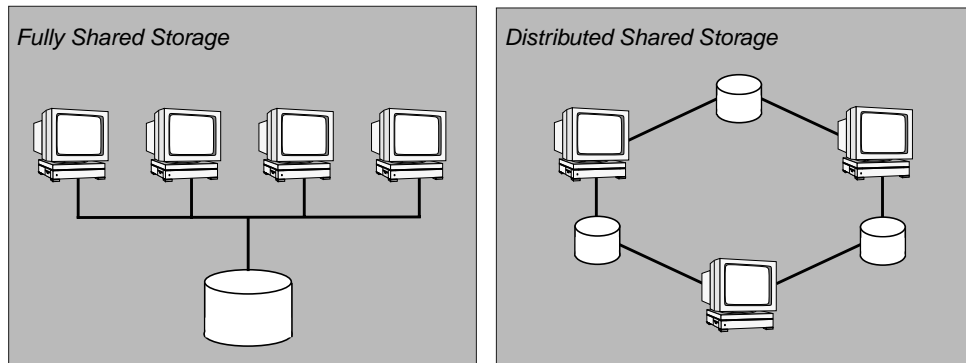




Shared Storage

A VCS hardware configuration consists of multiple systems that can be connected to shared storage via I/O channels. (See the accompanying installation guide for the maximum number of systems allowed in a hardware configuration.) Shared storage provides multiple systems a single access path to the same data source. It also enables VCS to restart applications on alternate systems when a system fails, thus ensuring high availability.

The figures below illustrate the flexibility of VCS shared storage configurations. (Note that VCS systems can only access storage that is physically attached.)



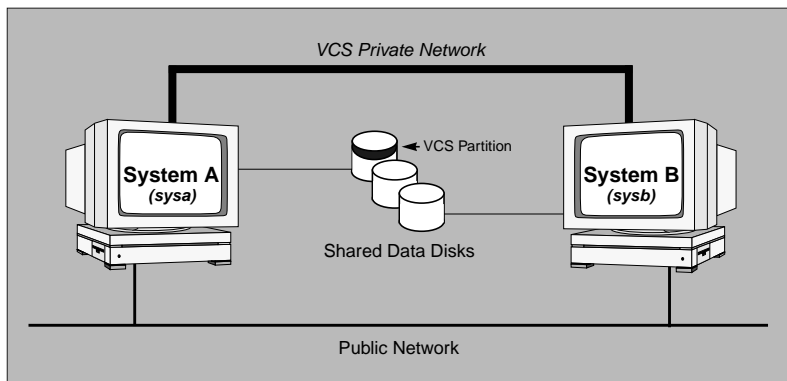
Two Examples of Shared Storage Configurations



Communication Channels

There are two types of channels available for VCS communication: networks and communication disks. Each cluster configuration requires at least two communication channels, and one of them must be a network channel. Remaining channels may be a combination of networks and communication disks. This requirement protects your cluster against network partitioning.

VCS supports a maximum of eight network channels and four communication disks. VCS communication disks must be connected to all systems in the cluster. We recommend configuring at least one disk on each I/O chain between the systems. (For configurations using communication disks, the cluster size is currently limited to eight systems.) Note that the term *disk* refers to a region of a physical disk partition, explained in the accompanying installation guide.



Two Systems Connected by a Single Ethernet and Communication Disk

Multiple Communication Channels

For multiple communication channels to be effective, they cannot fail simultaneously. By configuring disks and networks, you diversify communication pathways, and decrease the chances of simultaneous failure. Configuring a VCS communication disk is also an effective way to monitor the shared disk controllers on each system.

Configuration Options for Membership Services (GAB)

Service Group Heartbeat Disks (UNIX Only)

Even with two communication channels, VCS can partition if all channels fail simultaneously; for example, if power fails to all private network hubs and you have not configured a communication disk. To protect application data in this scenario, VCS offers another type of disk monitoring facility called *service group heartbeat disks*. Heartbeat disks are associated with service groups, and must be accessible from each system that can run the service group. The number of systems that can be supported by this feature is unlimited.

The heartbeat disk resource (ServiceGroupHB resource type) must be brought online before other resources in the group. Bringing this resource online first starts an internal process that periodically writes heartbeats to the disk. This involves incrementing a counter, which allows other systems to recognize that the ServiceGroupHB resource is online. Only one system can initiate this process. When attempting to bring a ServiceGroupHB resource online, VCS monitors the disk to detect if heartbeats are being written by another system. If they are, VCS does not bring the resource online, thereby stalling the process of bringing the service group online and preventing data corruption.

Disk Regions

Each VCS communication or heartbeat disk refers to a 64K (128-block) region of a physical disk partition. These regions are specified by a block device name and block offset. Multiple regions can be configured in different partitions on the same physical disk, and in the same partition at different, non-overlapping offsets. Communication and heartbeat disk regions can also coexist on a disk controlled by VERITAS Volume Manager.

A VCS communication disk requires two physical disk regions, one for *seeding* (described on page 184), and one for VCS. If there are multiple I/O chains connected to all systems, we recommend configuring a communication disk on each chain. Each disk requires two 128-block regions. If the partition begins in the first cylinder of the disk, avoid blocks 0-15 to leave room for the partition table.

Writing heartbeats to a service group heartbeat disk provides robust protection. Specifically, the heartbeat process can survive an abrupt shutdown of VCS, such as an administrative `kill` command or a VCS engine software error. Multiple heartbeat disks can be configured in the same ServiceGroupHB resource. All service group heartbeat disks must be available for the resource to go online, but only one disk is required for the resource to remain online.





The VCS configuration language specifies the makeup of service groups and their associated entities, such as resource types, resources, and attributes. These specifications are expressed in configuration files, the names of which contain the suffix `.cf`. For example, the main body of the configuration is in `main.cf`. Using an `include` statement, it references the file `types.cf`, which specifies resource types.

Configuration files reside on each system in the following locations:

- ◆ On **UNIX**: `/etc/VRTSvcs/conf/config`
- ◆ On **Windows NT** (default): `VCS_HOME\conf\config`

VCS is initialized by preparing these files on one system. VCS then propagates this information to each system in the cluster. There are three ways to generate these files:

- ✓ Use Cluster Manager or Configuration Editor. Cluster Manager is the graphical user interface for VCS. Configuration Editor is a configuration tool that enables you to generate new configuration files (`main.cf` and `types.cf`) while VCS is offline. (See page 77 for details.)
- ✓ If VCS is running, use the `haconf` command to dump the configuration information from the VCS engine. (This command launches the `hacf` utility, described on page 28.)
- ✓ If VCS is not running, use a text editor to create and modify the files.

In a VCS cluster, the first system to be brought online reads the configuration file and creates an internal representation of the configuration. Systems brought online after the first system derive their information from systems already running in the cluster. When a configuration is changed dynamically, or when a system joins a running cluster, the current configuration is written to the files `main.cf`, `types.cf`, and any files included by `main.cf`.



How the Configuration File Defines a Cluster

Listed below are the components of a VCS configuration file. Descriptions of each component begin on page 20.

- ◆ include clauses
- ◆ resource type definitions
- ◆ cluster definition
- ◆ system definitions
- ◆ snmp definition
- ◆ service group definitions
- ◆ resource definitions
- ◆ resource dependency clauses
- ◆ service group dependency clauses

About VCS Attributes

Most components in the VCS configuration contain attributes, and each attribute has a definition and a value. You define an attribute by specifying its data type and dimension (described on page 19). Attributes also have default values that are assigned when a value is not specified.

An attribute whose value applies to all systems is *global* in scope. An attribute whose value applies on a per-system basis is *local* in scope. The “at” operator (@) indicates the system to which a local value applies.

Attribute values of resource types and resources appear in the configuration file. Other attribute values are predefined by VCS and do not appear. Additionally, some attribute values are valid only while the VCS engine is running and are not written to the configuration file. See [Appendix C](#) for a complete list of the VCS attributes.

Valid attribute data types are *str* (string), *int* (integer), and *boolean*. Valid attribute dimensions are *scalar*, *vector*, *keylist*, and *association*. The following tables provide definitions of each.

Data Type	Description
string	A string is a sequence of characters enclosed by double quotes. A string may also contain double quotes, but the quotes must be immediately preceded by a backslash. A backslash is represented in a string as <code>\\</code> . Quotes are not required if a string begins with a letter, and contains only letters, numbers, dashes (-), and underscores (_).
integer	Signed integer constants are a sequence of digits from 0 to 9. They may be preceded by a dash, and are interpreted in base 10.
boolean	A boolean is an integer, the possible values of which are 0 (false) and 1 (true).

Dimension	Description
scalar	A scalar has only one value. This is the default dimension.
vector	A vector is an ordered list of values. Each value is indexed using a positive integer beginning with zero. A set of brackets ([]) denotes that the dimension is a vector. Brackets are specified after the attribute name on the attribute definition. For example, to designate a dependency between resource types specified in the service group list, and all instances of the respective resource type: <code>Dependencies[] = { Mount, Disk, DiskGroup }</code>
keylist	A keylist is an unordered list of strings, and each string is unique within the list. For example, to designate the list of systems on which a service group will be started with VCS (usually at system boot): <code>AutoStartList = { sysa, sysb, sysc }</code>
association	An association is an unordered list of name-value pairs. Each pair is separated by an equal sign. A set of braces ({}) denotes that an attribute is an association. Braces are specified after the attribute name on the attribute definition. For example, to designate the list of systems on which the service group is configured to run and the system's priorities: <code>StartList{} = { sysa=1, sysb=2, sysc=3 }</code>



Include Clause

Similar to the `#include` directive in C, the VCS `include` clause informs the `haconf` utility to get input from another file. The files to be included must reside in the following directories:

- ◆ On **UNIX**: `/etc/VRTSvcs/conf/config`
- ◆ On **Windows NT** (default): `VCS_HOME\conf\config`

Note Only the file `main.cf` can contain `include` clauses. Files included by `main.cf` cannot.

Resource Type Definitions

A resource type definition consists of:

- ◆ The keyword `type`.
- ◆ The name of the resource type. The name must begin with a letter, and must contain only letters, numbers, dashes (-), and underscores (_).
- ◆ The definition and default values of attributes associated with the resources of the resource type.

Resource attributes are classified as *type-independent* or *type-specific*. Type-independent resource attributes are common to resources of all types and are predefined by VCS. The default values of these attributes can be modified if required. Type-specific resource attributes are declared in the type definition (`types.cf`).

- ◆ The definition and default values of attributes associated with the resource type only. These are *static attributes*, the values of which are the same for all resources of the type. For example, a list of items passed to the agent is the same for all resources of a particular type. Static attributes are also classified as type-independent or type-specific.
- ◆ An optional *resource name rule* that specifies how to generate the unique name of a resource when the user does not specify one.

About the VCS Name Rule

Each VCS resource must have a unique name. You can specify this name when you define the resource, or it can be generated for you by the utility `hacf` using the resource type's `NameRule`. Resource names must begin with a letter, and contain only letters, numbers, dashes (-), and underscores (_). The following two samples show the complete generic definitions of Mount types on UNIX and Windows NT.

◆ On UNIX

```
type Mount (
    str MountPoint
    str BlockDevice
    str Type
    str MountOpt
    str FsckOpt
    static str ArgList[] = { MountPoint, BlockDevice,
        Type, MountOpt, FsckOpt }
    NameRule = resource.MountPoint
)
```

◆ On Windows NT

```
type Mount (
    static str ArgList[] = { DriveLetter, PartitionNo, FileSystemType,
        DiskResName, "DiskResName:DiskNo", "DiskResName:Signature",
        ForceUnmount, ListApplication }
    NameRule = MOUNT_ + resource.DriveLetter
    str DriveLetter
    int PartitionNo
    str FileSystemType
    str DiskResName
    int ForceUnmount
    int ListApplication = 1
)
```

In the preceding examples, the resource name would be generated from the value of the `MountPoint` resource attribute on UNIX, and the `DriveLetter` resource attribute on Windows NT. When generating a name, all special characters are replaced by an underscore, and leading underscores are removed. If the generated name still does not begin with a letter, it is prefixed by the name of the resource type followed by an underscore.

Name rules may also contain references to attributes of the resource's service group, and may consist of the concatenation of multiple attributes and string literals. The plus (+) operator denotes a concatenation.



The following sample defines the resource type NFS, and defines a NFS resource on **UNIX**:

```
type NFS (  
    int Nservers  
    NameRule = "NFS_" + group.Name + "_" + resource.Nservers  
    static str ArgList[] = { Nservers }  
    static str Operations = OnOnly  
)
```

If the following resource was a part of a group named "groupx":

```
NFS (  
    Nservers = 24  
)
```

The generated name would be NFS_groupx_24.

The next sample defines the resource type Lanman, and defines a Lanman resource on **Windows NT**:

```
type Lanman (  
    static str ArgList[] = { VirtualName }  
    NameRule = Lanman_ + resource.VirtualName  
    str VirtualName  
)
```

If the following resource was a part of a group named "groupz":

```
Lanman (  
    VirtualName = thor17  
)
```

The generated name would be Lanman_thor17.

Note An attribute used in the NameRule must be scalar in dimension and global in scope when the resource name does not appear.

Cluster Definitions

The cluster object contains attributes that apply to the entire cluster. A cluster definition consists of:

- ◆ The keyword `cluster`.
- ◆ The name of the cluster. The name must begin with a letter and must only contain letters, numbers, dashes (-), and underscores (_).
- ◆ The cluster's attribute values enclosed in parentheses. Cluster attributes are predefined by VCS.

System Definitions

The system object contains attributes that apply to each system. A system definition consists of:

- ◆ The keyword `system`.
- ◆ The name of the system. The name must begin with a letter and must only contain letters, numbers, dashes (-), and underscores (_).
- ◆ The system's attribute values enclosed in parentheses. System attributes are predefined by VCS.

Note Before adding a system to the `main.cf` file, you must first add the system to the `LLTHOSTS` file on each system in the cluster.



Domain-Qualified System Names—UNIX Only

VCS does not accept fully qualified domain names for its configuration. Specifically, if the name contains periods it must be changed, or VCS must be configured to use a different name. System names must be unique on each system in the cluster.

▼ To change a system name

1. Type `uname -S` at the command prompt.
2. Enter the new system name.

▼ To configure VCS to use a different name

1. Type `uname -n` at the command prompt.
2. If the system name is `sysa.acme.com`, configure VCS to use the name `sysa` only.
 - a. Create the file `/etc/VRTSvcs/conf/sysname`. This file contains the name that VCS uses for the local system:

```
# cat /etc/VRTSvcs/conf/sysname
```
 - b. Repeat [step a](#) on all systems using non-domain-qualified system names prior to starting VCS.

SNMP Definitions

The SNMP object contains attributes that describe traps sent to an SNMP console for events in VCS. An SNMP definition consists of:

- ◆ The keyword `snmp`.
- ◆ The name of the SNMP. The name must begin with a letter and must only contain letters, numbers, dashes (-), and underscores (_).
- ◆ The SNMP's attribute values enclosed in parentheses. SNMP attributes are predefined by VCS.

For example, an SNMP definition would resemble:

```
snmp vcs (  
    Enabled = 1  
    IPAddr = "166.98.13.202"  
    TrapList = {  
        1 = "A new system has joined a VCS Cluster",  
        2 = "An existing system has changed its state",  
        3 = "A service group has changed its state",  
        4 = "One or more heartbeat links has gone down",  
        5 = "An HA service has done a manual restart",  
        6 = "An HA service has been manually idled",  
        7 = "An HA service has been successfully failed  
            over to a backup system"  
    }  
    Port = 162  
)
```



Service Group Definitions

A service group definition consists of:

- ◆ The keyword `group`.
- ◆ The name of the service group. The name must begin with a letter and must only contain letters, numbers, dashes (-), and underscores (_).
- ◆ The service group's attribute values enclosed in parentheses. Service group attributes are predefined by VCS (described below).
- ◆ The definition of all service group resources.
- ◆ The dependencies between the service group resources.
- ◆ The dependency of the service group on other service groups (described in “[What is a Service Group Dependency?](#)” on page 129).

Service Group Attributes

A service group is configured to run on a specific set of systems in the cluster. Attributes are local or global regardless of their dimension. If a vector attribute is local, that attribute will have a vector for each system.

For example, a service group definition would resemble:

```
group groupx (  
    SystemList = { sysa, sysb }  
    AutoStartList = { sysa }  
)
```

Resource Definitions

A resource belongs to a single service group. If an entity is in multiple service groups it must be represented as multiple VCS resources, each within a separate service group.

A resource definition consists of:

- ◆ The resource type.
- ◆ An optional name. The name must begin with a letter, and must only contain letters, numbers, dashes (-), and underscores (_). If a name is not specified explicitly, the utility `hacf` generates a name, based on the `NameRule` of the resource type. If the name is omitted, all attributes that appear in the `NameRule` must be global in scope. (See “[About the VCS Name Rule](#)” on page 21.)
- ◆ The resource's type-independent and type-specific attribute values enclosed in parentheses.

Resource Attributes

The scope of resource attributes can be local or global. The attribute values passed as arguments to agent entry points are values local to the system on which the agent is running.

For example, a Mount resource definition with an explicit name on **UNIX** would resemble:

```
Mount export1 (
    MountPoint = "/export1"
    BlockDevice = "/dev/sharedg/voloracle"
    Type = vxfs
    MountOpt = rw
    FsckOpt = ""
)
```

A Mount resource definition with an explicit name on **Windows NT** would resemble:

```
Mount MyMount (
    DriveLetter = W
    PartitionNo = 2
    FileSystemType = NTFS
    DiskResName = MyDisk
    ForceUnmount = 1
    ListApplication = 1
)
```

Resource Dependencies

A dependency between resources is indicated by the keyword `requires` between two resource names. This indicates that the second resource (the child) must be online before the first resource (the parent) can be brought online. Conversely, the parent must be offline before the child can be taken offline. Also, faults of the children are propagated to the parent. This is the most common resource dependency. For example:

```
export1 requires sharedg_voloracle
```



Managing the VCS Configuration File

The hacf Utility

The `hacf` utility translates the VCS configuration language into a syntax that can be read by the VCS engine. Specifically, `hacf` translates the contents of the main configuration file, `main.cf`, into commands for the VCS server. You can use `hacf` to verify (check syntax) of `main.cf` and the type definition file, `types.cf`. VCS does not execute if `hacf` detects errors in the configuration. No error message and a return value of zero indicates that the syntax is legal.

Verifying a Configuration

To verify a configuration, type:

```
hacf -verify config_directory
```

The variable `config_directory` refers to directories containing a `main.cf` file and any `.cf` files included in `main.cf`.

Generating a main.cmd file from main.cf

The `hacf` utility also enables you to generate a `main.cmd` file from the `main.cf`.

To generate a `main.cmd` file from the `main.cf` configuration file, type:

```
hacf -cftocmd config_directory [-dest dest_dir] [-display]
```

The variable `config_directory` refers to directory in which the `main.cf` file resides. The variable `dest_dir` refers to the file in which the `main.cmd` file is generated (“destination directory”). If the `-dest` option is not specified, the `main.cmd` file is generated in the same directory as `main.cf` (typically `/etc/VRTSvcs/conf/config`). If the `-display` option is indicated, the output of the command is displayed.

Generating a main.cf file from main.cmd

Conversely, to generate a `main.cf` file from the `main.cmd` file, type:

```
hacf cmdtocf config_directory [-dest dest_dir]
```

The variable `config_directory` refers to directory in which the `main.cmd` file resides. The variable `dest_dir` refers to the file in which the `main.cf` file is generated. If the `-dest` option is not specified, the `main.cf` file is generated in the same directory as `main.cmd`.

The `hacf` utility can be used on any directory; however, VCS looks for the configuration file in the following directories:

- ◆ On **UNIX**: `$VCS_CONF/conf/config`

If you create the configuration file in another location, you must copy it into `/etc/VRTSvcs/conf/config`

- ◆ On **Windows NT** (default): `VCS_HOME\conf\config`

If you create the configuration file in another location, you must copy it into the above directory.

Note Multiple copies of the `hacf` utility cannot run on the same configuration directory simultaneously. A newly invoked `hacf` will exit if it detects that another copy is running.

Loading a Configuration

The `hacf` utility automatically verifies the configuration before loading it into VCS. The configuration is not loaded under the following conditions:

- ◆ If `main.cf` or `include` files are missing.
- ◆ If syntax errors appear in the `.cf` files.
- ◆ If the configuration file is marked “stale.” A `.stale` file is created in the configuration directory when you indicate that you intend to change a running configuration. See “[Configuration Read/Write and Read-Only Modes](#)” on page 63 for details.

Dumping a Running Configuration

A configuration is dumped (written to disk) when you indicate that you have finished changing it. The configuration is also dumped on a system when the system joins the VCS cluster. When VCS dumps a running configuration, it is always pretty-printed. VCS removes the `.stale` file following a successful dump.

Multiple Versions of `.cf` Files

When `hacf` creates a `.cf` file, it does *not* overwrite existing `.cf` files. A copy of the file remains in the directory, and its name includes a suffix of the date and time it was created it, such as `main.cf.03Dec1999.175904`, and a symbolic link from the original name. In addition, the previous version of any `.cf` file is saved with the suffix `.previous`; for example, `main.cf.previous`.



Sample Configuration Files

The file `main.cf` is where you specify the application dependency graphs and the resources to be controlled by VCS. This file is read by VCS at startup. The `types.cf` file is where you specify the resource type definitions. This file is read by `main.cf` via the `include` clause. See page 33 for the UNIX sample `types.cf` file. See page 39 for the Windows NT sample `types.cf` file.

Sample `main.cf` File for UNIX

The following sample consists of two systems, HA1 and HA2, and a service group, Sybase.

```
include "types.cf"
include "SybaseTypes.cf"

cluster HA (
    UserNames = { root = cD9YoocT4eFao }
    CounterInterval = 5
    Factor = { runque = 5, memory = 1, disk = 10, cpu = 25,
              network = 5 }
    MaxFactor = { runque = 100, memory = 10, disk = 100, cpu = 100,
                  network = 100 }
)

system HA1
system HA2

snmp HA (
    TrapList = { 1 = "A new system has joined the VCS Cluster",
                  2 = "An existing system has changed its state",
                  3 = "A service group has changed its state",
                  4 = "One or more heartbeat links has gone down",
                  5 = "An HA service has done a manual restart",
                  6 = "An HA service has been manually idled",
                  7 = "An HA service has been successfully started" }
)

group Sybase (
    SystemList = { HA1, HA2 }
)

    DiskGroup sybase_dg (
        DiskGroup = sybase
    )
```

```
IP sybase_ip (
    Device = hme0
    Address = "166.98.21.114"
)

Mount sybase_mount (
    MountPoint = "/opt/apps"
    BlockDevice = "/dev/vx/dsk/sybase/sybase_install"
    FSType = vxfs
)

NIC sybase_hme0 (
    Device = hme0
    NetWorkType = ether
)

Sybase sybase_prod (
    Server = vcs2
    Owner = sybase
    Home = "/opt/apps"
    Version = sybasell
    SA = sa
    SApswd = "VCSSY: /sapword"
)

SybaseBk sybase_bk (
    Backupserver = vcs2_back
    Owner = sybase
    Home = "/opt/apps/"
    Version = "11.9.2"
    Server = vcs2
    SA = sa
    SApswd = "VCSSY: /sapword"
)

sybase_ip requires sybase_hme0
sybase_mount requires sybase_dg
sybase_prod requires sybase_ip
sybase_bk requires sybase_prod
```



```
// resource dependency tree
//
//   group Sybase
//   {
//     SybaseBk sybase_bk
//     {
//       Sybase sybase_prod
//       {
//         Mount sybase_mount
//         {
//           DiskGroup sybase2_dg
//         }
//       }
//     }
//   }
//   IP sybase_ip
//   {
//     NIC sybase_hme0
//   }
// }
// }
```


Sample types.cf File for UNIX

```
type ElifNone (
    static str ArgList[] = { PathName }
    NameRule = resource.PathName
    static str Operations = None
    str PathName
)

type FileNone (
    static str ArgList[] = { PathName }
    NameRule = resource.PathName
    static str Operations = None
    str PathName
)

type FileOnOff (
    static str ArgList[] = { PathName }
    NameRule = resource.PathName
    str PathName
)

type FileOnOnly (
    static str ArgList[] = { PathName }
    NameRule = resource.PathName
    static str Operations = OnOnly
    str PathName
)

type IP (
    static str ArgList[] = { Device, Address, NetMask, Options,
                          ArpDelay, IfconfigTwice }
    NameRule = IP_ + resource.Address
    str Device
    str Address
    str NetMask
    str Options
    int ArpDelay = 1
    int IfconfigTwice = 0
)
```



```
type IPMultiNIC (
    static str ArgList[] =
        { "MultiNICResName:Device", Address, NetMask,
          "MultiNICResName:ArpDelay", Options,
          "MultiNICResName:Probed", MultiNICResName,
          IfconfigTwice }
    NameRule = IPMultiNIC_ + resource.Address
    str Address
    str NetMask
    str Options
    str MultiNICResName
    int IfconfigTwice = 0
)

type LVMLogicalVolume (
    static str ArgList[] = { LogicalVolume, VolumeGroup }
    NameRule = resource.VolumeGroup + "_" +
    resource + LogicalVolume
    str LogicalVolume
    str VolumeGroup
)

type LVMVolumeGroup (
    static str ArgList[] = { VolumeGroup }
    NameRule = resource.VolumeGroup
    str VolumeGroup
)

type Mount (
    static str ArgList[] = { MountPoint, BlockDevice, Type,
        MountOpt, FsckOpt }
    NameRule = resource.MountPoint
    str MountPoint
    str BlockDevice
    str FSType
    str MountOpt
    str FsckOpt
)
```



```

type MultiNICA (
    static str ArgList[] = { Device, NetMask, ArpDelay, Options,
        RouteOptions, MonitorOnly, IfconfigTwice,
        HandshakeInterval, NetworkHosts }
    static int MonitorTimeout = 120
    NameRule = MultiNICA_ + group.Name
    static str Operations = None
    str Device{}
    str NetMask
    int ArpDelay = 1
    str Options
    str RouteOptions
    int IfconfigTwice = 0
    int HandshakeInterval = 0
    str NetworkHosts []
)

type NFS (
    static int RestartLimit = 1
    static str ArgList[] = { Nservers }
    NameRule = NFS_ + group.Name + "_" + resource.Nservers
    static str Operations = OnOnly
    int Nservers = 4
)

type NIC (
    static str ArgList[] = { Device, NetworkType, NetworkHosts }
    NameRule = group.Name + "_" + resource.Device
    static str Operations = None
    str Device
    str NetworkType
    str NetworkHosts[]
)

type Phantom (
    static str ArgList[] = {}
    NameRule = Phantom_group.Name
)

type Process (
    static str ArgList[] = { PathName, Arguments }
    NameRule = Resource.PathName
    str PathName
    str Arguments
)

```



```
type Proxy (
    static str ArgList[] = { TargetName, TargetSysName,
        "TargetResName:Probed", "TargetResName:State" }
    NameRule = { Proxy_+resource.TargetResName }
    static str Operations = None
    str TargetResName
    str TargetSysName
)

type ServiceGroupHB (
    static str ArgList[] = { Disks }
    NameRule = SGHB_ + resource.Disks
    str Disks[]
)

type Share (
    static str ArgList[] = { PathName, Options,
        OnlineNFSRestart, OfflineNFSRestart }
    NameRule = nfs + resource.PathName
    str PathName
    str Options
    int OnlineNFSRestart = 0
    int OfflineNFSRestart = 1
)
```

Sample main.cf File for Windows NT

```
include "types.cf"

cluster VCLUSTER (
  UserNames = { }
  CounterInterval = 5
  Factor = { runque = 5, memory = 1, disk = 10, cpu = 25,
            network = 5 }
  MaxFactor = { runque = 100, memory = 10, disk = 100, cpu = 100,
              network = 100 }
)

system VCSNT5

system VCSNT6

snmp VCLUSTER (
  TrapList = { 1 = "A new system has joined the VCS Cluster",
              2 = "An existing system has changed its state",
              3 = "A service group has changed its state",
              4 = "One or more heartbeat links has gone down",
              5 = "An HA service has done a manual restart",
              6 = "An HA service has been manually idled",
              7 = "An HA service has been successfully started" }
)

group VCS_Sample_Conf (
  SystemList = { VCSNT6, VCSNT5 }
  PrintTree = 0
  AutoStart = 0
)

Disk DISK_1 (
  DiskNo = 1
  Signature = 4191935953
)

Lanman LEEDS (
  VirtualName = LEEDS
)

Mount MOUNT_X (
  DriveLetter = X
  PartitionNo = 1
  FileSystemType = NTFS
  DiskNo = 1
  Signature = 4191935953
  ForceUnmount = 1
)
```



```
FileShare FS_USER1 (  
    Critical = 0  
    ShareName = USER1  
    MaxUsers = 200  
    PathName = "\\HOME\\USER1"  
    MountResName = MOUNT_X  
)  
  
IP IP_172_16_1_32 (  
    Address = "172.16.1.32"  
    SubNetMask = "255.255.255.0"  
    AdapterName @VCSNT6 = SMCPWR1  
    AdapterName @VCSNT5 = ADPTSF1  
)  
  
NIC NIC_TEST (  
    AdapterName @VCSNT6 = SMCPWR1  
    AdapterName @VCSNT5 = ADPTSF1  
    UseConnectionStatus = 1  
)  
  
LEEDS requires FS_USER1  
FS_USER1 requires MOUNT_X  
LEEDS requires IP_172_16_1_32  
IP_172_16_1_32 requires NIC_TEST  
LEEDS requires MOUNT_X  
MOUNT_X requires DISK_1
```

Sample types.cf File for Windows NT

```
type CompositeFileShare (
    static str ArgList[] = { AgentDebug, MaxUsers,
        "MountResName:DriveLetter", PathAndShareName }
    NameRule = FileShare_ + resource.ShareName
    str PathAndShareName{}
    str MaxUsers
    str MountResName
    int AgentDebug = 1
)

type Disk (
    static str ArgList[] = { DiskNo, Signature, AgentDebug }
    static str Operations = None
    NameRule = DISK_ + resource.DiskNo
    int DiskNo
    str Signature
    int AgentDebug = 0
)

type DiskRes (
    static str ArgList[] = { Disks, Signatures, Weights, FailFast,
        GracePeriod, MaxKernelThreads, ReserveAll, AgentDebug }
    int Disks[]
    str Signatures[]
    int Weights[]
    int FailFast = 1
    int GracePeriod = 5
    int MaxKernelThreads = 5
    boolean ReserveAll = 0
    boolean AgentDebug = 0
)
```



```
type ElifNone (
    static str ArgList[] = { PathName, AgentDebug }
    NameRule = resource.PathName
    static str Operations = None
    str PathName
    int AgentDebug = 0
)

type FileNone (
    static str ArgList[] = { PathName, AgentDebug }
    NameRule = resource.PathName
    static str Operations = None
    str PathName
    int AgentDebug = 0
)

type FileOnOff (
    static str ArgList[] = { PathName, AgentDebug }
    NameRule = resource.PathName
    str PathName
    int AgentDebug = 0
)

type FileOnOnly (
    static str ArgList[] = { PathName, AgentDebug }
    NameRule = resource.PathName
    static str Operations = OnOnly
    str PathName
    int AgentDebug = 0
)

type Exchange (
    static str ArgList[] = { "LanmanResName:VirtualName",
        ExchangePath, InternetMailService, DirSynchronization,
        MailConnectorInterchange, AgentDebug }
    NameRule = EXCH_ + resource.LanmanResName
    str LanmanResName
    str ExchangePath
    int InternetMailService = 0
    int DirSynchronization = 0
    int MailConnectorInterchange = 0
    static int OnlineTimeout = 600
    int AgentDebug = 0
)
```



```
type PrintShare (
    static str ArgList[] = { PrinterName, ShareName, MaxUsers,
        "MountResName:DriveLetter", AgentDebug }
    NameRule = PrintShare_ + resource.ShareName
    str ShareName
    int MaxUsers
    str PrinterName
    str MountResName
    int AgentDebug = 0
)

type FileShare (
    static str ArgList[] = { PathName, ShareName, MaxUsers,
        "MountResName:DriveLetter", AgentDebug }
    NameRule = FileShare_ + resource.ShareName
    str ShareName
    int MaxUsers
    str PathName
    str MountResName
    int AgentDebug = 0
)

type GenericService (
    static str ArgList[] = { ServiceName, DelayAfterOnline,
        DelayAfterOffline, service_arg, AgentDebug }
    NameRule = resource.ServiceName
    int AgentDebug = 0
    str ServiceName
    int DelayAfterOnline = 10
    int DelayAfterOffline = 10
    str service_arg[]
)

type IIS (
    static str ArgList[] = { FTPService, FTPPort, FTPTimeOut,
        WWWService, WWWPort, WWWTimeOut, SMTPService,
        SMTPPort, SMTPTimeOut, NNTPService, NNTPPort,
        NNTPTimeOut, DetailMonitor, AgentDebug }
    NameRule = IIS_ + group.Name
    int FTPService = 1
    int FTPPort = 21
    int FTPTimeOut = 10
    int WWWService = 1
    int WWWPort = 80
    int WWWTimeOut = 10
    int SMTPService = 0
    int SMTPPort = 25
    int SMTPTimeOut = 10
)
```



```
    int NNTPService = 0
    int NNTPPort = 119
    int NNTPTimeOut = 10
    int DetailMonitor = 1
    int AgentDebug = 0
)

type IP (
    static str ArgList[] = { Address, SubNetMask, AdapterName,
                           AgentDebug }
    NameRule = IP_ + resource.Address
    str Address
    str SubNetMask
    str AdapterName
    int AgentDebug = 0
)

type Lanman (
    static str ArgList[] = { VirtualName, AgentDebug }
    NameRule = Lanman_ + resource.VirtualName
    str VirtualName
    int AgentDebug = 0
)

type Mount (
    static str ArgList[] = { DriveLetter, PartitionNo,
                           FileSystemType, DiskNo, Signature, ForceUnmount,
                           ListApplication, AgentDebug }
    NameRule = MOUNT_ + resource.DriveLetter
    str DriveLetter
    int PartitionNo
    str FileSystemType
    int DiskNo
    str Signature
    boolean ForceUnmount = 0
    boolean ListApplication = 1
    int AgentDebug = 0
)
```

```

type NIC (
    static str ArgList[] = { AdapterName, PingTimeoutMseconds,
                            MaxTxErrorPercentage, MaxTxErrInterval,
                            UseConnectionStatus, PingHostList, AgentDebug}
    NameRule = NIC_ + resource.AdapterName
    static str Operations = None
    str AdapterName
    int PingTimeoutMseconds = 1000
    int MaxTxErrorPercentage = 50
    int MaxTxErrInterval = 10
    int UseConnectionStatus = 0
    str PingHostList[]
    int AgentDebug = 0
)

type LotusDomino(
    static str ArgList[] = { DelayAfterOnline, DelayAfterOffline,
                            DetailMonitor, ServerName, DbFileName, AgentDebug }
    NameRule = LotusDomino
    int DelayAfterOnline = 180
    int DelayAfterOffline = 180
    int DetailMonitor = 0
    str ServerName
    str DbFileName = log
    int AgentDebug = 0
)

type MultiNICA (
    static str ArgList[] = { AdapterList, PingTimeoutMseconds,
                            MaxTxErrorPercentage, MaxTxErrInterval,
                            UseConnectionStatus, AdminIPAddr, AdminSubnetMask,
                            PingHostList, AgentDebug }
    NameRule = MultiNICA_ + group.Name
    static str Operations = None
    str AdapterList[]
    int PingTimeoutMseconds = 1000
    int MaxTxErrorPercentage = 50
    int MaxTxErrInterval = 10
    int UseConnectionStatus = 0
    str AdminIPAddr
    str AdminSubnetMask
    str PingHostList[]
    int AgentDebug = 0
)

```



```
type IPMultiNIC (
    static str ArgList[] = { "MultiNICAResName:AdapterList",
        Address, SubNetMask, AgentDebug }
    NameRule = IPMultiNIC_ + resource.Address
    str MultiNICAResName
    str Address
    str SubNetMask
    int AgentDebug = 0
)

type Phantom (
    static str ArgList[] = { AgentDebug }
    NameRule = Phantom_ + group.Name
    int AgentDebug = 0
)

type Proxy (
    static str ArgList[] = { TargetResName, TargetSysName,
        "TargetResName:Probed", "TargetResName:State",
        AgentDebug }
    NameRule = Proxy_ + resource.TargetResName
    static str Operations = None
    str TargetResName
    str TargetSysName
    int AgentDebug = 0
)

type SQLServer (
    static str ArgList[] = { ServiceName, DelayAfterOnline,
        DelayAfterOffline, service_arg }
    NameRule = SQLServer
    str ServiceName
    int DelayAfterOnline = 10
    int DelayAfterOffline = 10
    str service_arg[]
)
)
```

Administering VCS From the Command Line

4

Most of the commands listed in this chapter can be entered from any system in the cluster but only when VCS is running. The command to start VCS is typically invoked at system startup. Instructions on how to start VCS on UNIX and Windows NT platforms begin on page 47.

VCS Environment Variables

Variable	Definition	Default
VCS_CONF	Root directory for VCS configuration files.	<ul style="list-style-type: none">◆ On UNIX: /etc/VRTSvcs◆ On Windows NT: <i>Install Drive</i>: \Program Files\VERITAS\cluster server Note If this variable is added or modified you must reboot the system to apply the changes.
VCS_HOME	Root directory for VCS executables.	<ul style="list-style-type: none">◆ On UNIX: /opt/VRTSvcs◆ On Windows NT: <i>Install Drive</i>: \Program Files\VERITAS\cluster server
VCS_GAB_PORT	GAB port to which VCS connects.	h Default is same for UNIX and Windows NT.
VCS_GAB_TIMEOUT	Timeout in milliseconds for VCS engine to send heartbeats to GAB.	15000 Default is same for UNIX and Windows NT. Note that if the specified timeout is exceeded, GAB kills the VCS engine, and all active service groups on system are disabled.



Variable	Definition	Default
VCS_HAD_RESTART_TIMEOUT	User must set this variable to designate the amount of time the hashadow process waits ("sleep time") before restarting HAD.	0 Default is same for UNIX and Windows NT.
VCS_LOG	Root directory for log files and temporary files.	<ul style="list-style-type: none"> ◆ On UNIX: /var/VRTSvcs ◆ On Windows NT: <i>Install Drive:\Program Files\VERITAS\cluster server</i> <p>Note If this variable is added or modified you must reboot the system to apply the changes.</p>
VCS_PORT	Port number of the VCS engine.	14141 Default is same for UNIX and Windows NT.
VCS_TEMP_DIR	Directory in which temporary information required by, or generated by, hacf is stored.	<p>Same default values as VCS_LOG:</p> <ul style="list-style-type: none"> ◆ On UNIX: /var/VRTSvcs ◆ On Windows NT: <i>Install Drive:\Program Files\VERITAS\cluster server</i> <p>Note This directory is created in /tmp on UNIX, and \ in Windows NT under the following conditions:</p> <ul style="list-style-type: none"> ◆ The variable is not set. ◆ The variable is set but the directory to which it is set does not exist. ◆ The utility hacf cannot locate the default locations.



How VCS Identifies the Local System

- ◆ On **UNIX**: VCS checks `$VCS_CONF/conf/sysname`
If this file does not exist, the local system is identified by its node name. To view the system's node name, type the command: `uname -n`
- ◆ On **Windows NT**: VCS uses the system's node name. To view the system's node name, type the command: `hostname`

To view the Windows NT system's node name from the desktop:

1. Left-click the Network Neighborhood icon.
2. Right-click the icon to display the drop-down menu.
3. Select Properties. The name of the system is listed in the Identification tab.

Starting VCS

When VCS is started on a system, and when that system is the only one running, VCS retrieves the configuration from the local configuration directory.

- ◆ On **UNIX**: `$VCS_CONF/conf/config`
- ◆ On **Windows NT** (default): `VCS_HOME\conf\config`

If the local configuration is valid, the VCS engine performs a `LOCAL_BUILD`, and the system transitions to the state of `RUNNING`, its normal operational state. If the local configuration is missing, invalid, or designated "stale," the system transitions to the state of `STALE_ADMIN_WAIT`, and the VCS engine waits for manual intervention, or for VCS to be started on a system that has a valid configuration.

If VCS is started on a system when other systems are already running VCS, the engine processes exchange their operational states according to the following conventions:

- ◆ If a system running VCS is in the state of `RUNNING`, the system joining the cluster performs a `REMOTE_BUILD` from that system and transitions to the state of `RUNNING`.
- ◆ If a system running VCS is in the state of `LOCAL_BUILD`, the system joining the cluster waits for that system to transition to `RUNNING`. It then performs a `REMOTE_BUILD` from that system and transitions to the state of `RUNNING`.
- ◆ If all systems running VCS are in the state of `STALE_ADMIN_WAIT`, and if the local configuration file of the system joining the cluster is valid, the joining system performs a `LOCAL_BUILD` and transitions to `RUNNING`. The other systems then perform `REMOTE_BUILDS` from the new system and transition to `RUNNING`.



- ◆ If all systems running VCS are in the state of `STALE_ADMIN_WAIT`, and if the local configuration file of the system joining the cluster is invalid, then the joining system also transitions to `STALE_ADMIN_WAIT`.

Note See Appendix B for a complete list of VCS system states and transitions.

Starting VCS on UNIX

The command to start VCS on UNIX is invoked from the file `/etc/rc3.d/S99vcs` or `/sbin/rc3.d/S99vcs`.

Type the following command to start VCS:

```
# hastart [-stale|-force]
```

Note that `-stale` and `-force` are optional. The option `-stale` instructs the engine to treat the local configuration as stale even if it is valid. The option `-force` instructs the engine to treat a stale, but otherwise valid, local configuration as valid.

If all systems are in `STALE_ADMIN_WAIT` or `ADMIN_WAIT`, enter the following command from any system in the cluster to force VCS to use the configuration file from the system specified by the variable `system`:

```
# hasys -force system
```

Starting VCS on Windows NT

VCS on Windows NT is started automatically during the installation procedure. If VCS fails to start, follow the instructions below to start VCS manually.

▼ From the Windows NT Services Applet

1. Double-click the Services icon in the Control Panel.
2. Select VERITAS High Availability Engine.
3. Click Start.
4. Click Close.

▼ From the Command Line

```
C:\> net start had
```

or

```
C:\> hastart
```



To start Command Server, type:

```
C:\> net start cmdserver
```

Note To specify command-line arguments to HAD, you must use the Windows NT Services applet. Enter the arguments in the Startup Parameters text field.

Stopping VCS

The `hastop` command stops the VCS engine and related processes. This command includes the following options:

```
hastop -local [-force | -evacuate]
hastop -sys system [-force | -evacuate]
hastop -all [-force]
```

The option `-evacuate`, when combined with `-local` or `-sys`, migrates the system's active service groups to another system in the cluster, before the system is stopped.

The option `-local` stops the engine on the system you typed the command.

The option `-sys` stops the engine on the system you specified.

The option `-all` stops the engine on all systems in the cluster.

The option `-force` allows the engine to be stopped without offlining service groups on that system.

When the VCS engine stops on a system, the system's service groups are also stopped and the group-level attribute `IntentOnline` for such groups is set to 0. Specifying the `-force` option does not bring down the service groups and does not affect the value of `IntentOnline`.

Note Stopping VCS on a system autodisables all service groups that include the system in their `SystemList` attribute. (This doesn't apply to systems that are powered off.) Note also that if you use the `-evacuate` option, evacuation occurs before VCS is brought down.

For UNIX Users

If using the command `reboot`, behavior is controlled by `ShutdownTimeOut` parameter. (See the *VERITAS Cluster Server Agent Developer's Guide* for more information on parameters.)



Additional Considerations for Stopping VCS

When VCS is stopped by options other than `-force` on a system with online service groups, the groups running on the system are taken offline and remain offline. This is indicated by VCS setting the attribute `IntentOnline` to 0. Using the option `-force` enables service groups to continue running while the engine is brought down and restarted (`IntentOnline` remains unchanged).

If VCS was restarted on the system and the system joined the rest of the cluster, the engine attempts to bring the service group back online on the systems listed in the group's `AutoStartList`.

Adding, Modifying, and Deleting Users

Guidelines

- ✓ The VCS configuration must be in read/write mode (described below).
- ✓ You may add, modify, and delete users on any system in the cluster (described on page 51).
- ✓ You must add users to the VCS configuration to monitor and administer VCS from the GUI (described on page 79).

Configuration Read/Write and Read-Only Modes

The commands to add, modify, and delete a user change the attribute values stored in the `.cf` files. Therefore, these commands can be executed only as `root` on UNIX and “Administrator” on Windows NT, and only if the VCS configuration is in read/write mode.

To set the mode to read/write, type the following command from any system in the cluster:

```
haconf -makerw
```

In addition to setting the configuration to read/write, this command designates the configuration stale by creating the following file on all systems running VCS:

- ◆ On **UNIX**: `$VCS_CONF/conf/config/.stale`
- ◆ On **Windows NT** (default): `VCS_HOME\config\config\stale`

When you have completed adding, modifying, and deleting users, reset the configuration to read-only:

```
haconf -dump -makero
```

In addition to setting the configuration to read-only, this command writes, or “dumps,” the configuration to disk and removes the configuration’s designation of stale.

Note You may perform the following procedures on any system in cluster.

▼ **To add a user with superuser access**

1. Set the configuration to read/write mode:

```
haconf -makerw
```

2. Add the user:

```
hauser -add user_name
```

3. Enter a password when prompted.

4. Reset the configuration to read-only:

```
haconf -dump -makero
```

▼ **To modify a user**

1. Set the configuration to read/write mode:

```
haconf -makerw
```

2. Modify the user:

```
hauser -modify user_name
```

3. Enter a new password when prompted.

4. Reset the configuration to read-only:

```
haconf -dump -makero
```



▼ To delete a user

1. Set the configuration to read/write mode:

```
haconf -makerw
```

2. Delete the user from the list of registered users:

```
hauser -delete user_name
```

3. Reset the configuration to read-only:

```
haconf -dump -makero
```

▼ To display a user

Type the following command to display a list of users and their encrypted passwords:

```
hauser -display [user_name]
```

If *user_name* is not specified, all users are displayed.

Querying VCS

VCS enables you to query information regarding various configuration entities, including resources, service groups, systems, resource types, agents, and clusters. You may enter query commands from any system in the cluster. Commands to display information on the VCS configuration or system states can be executed by all users. You do not need `root` or Administrator privileges.

Query commands are described in the following sections, beginning with resources.

Resources

For a list of all resources whose values match given conditional statement, type:

```
hares -list [conditional]
```

Conditional statements can take three forms:

```
Attribute=Value  
Attribute!=Value  
Attribute=~Value
```

Multiple conditional statements may be used and imply AND logic. If no conditional statement is specified, all resources in the cluster are listed.

For a list of a resource's dependencies, type:

```
hares -dep [resource]
```

For information about a particular resource, type:

```
hares -display [resource]
```

If *resource* is not specified, information regarding all resources is displayed.

To display details of a particular attribute, type:

```
hares -display -attribute [attribute]
```

To confirm that an attribute's values are the same on all systems, type:

```
hares -global
```

To display resources of a particular service group, type:

```
hares -display -group [service_group]
```

To display resources of a particular resource type, enter:

```
hares -display -type [resource_type]
```

To display attributes of a particular system, type:

```
hares -display -sys [system]
```



Service Groups

For a list of all service groups whose values match given conditional statement, type:

```
hagrp -list [conditional]
```

Conditional statements can take three forms:

```
Attribute=Value  
Attribute!=Value  
Attribute=~Value
```

Multiple conditional statements may be used and imply AND logic. If no conditional statement is specified, all service groups in the cluster are listed.

To clear faulted, non-persistent resources in a specific service group, type:

```
hagrp -clear [service_group] -sys [system]
```

Clearing a resource automatically initiates the online process that was previously blocked while waiting for the resource to become clear.

If *system* is specified, all faulted, non-persistent resources are cleared from that system only. If *system* is not specified, the service group is cleared on all systems in the group's SystemList in which at least one non-persistent resource has faulted.

To display the current state of specific service group on a specific system, type:

```
hagrp -state [service_group] -sys [system]
```

For a list of a service group's resources, type:

```
hagrp -resources [service_group]
```

For a list of a service group's dependencies, type:

```
hagrp -dep [service_group]
```

For information about service groups on a particular system, type:

```
hagrp -display [service_group]
```

If *service_group* is not specified, information regarding all service groups is displayed.

To display attributes of a particular system, type:

```
hagrp -display [service_group] [-attribute attribute]  
[-sys system]
```

Systems

For a list of systems in the cluster, type:

```
hasys -list
```

For information about each system, type:

```
hasys -display [system]
```

Resource Types

For a list of resource types, enter:

```
hatype -list
```

For a list of all resources of a particular type, enter:

```
hatype -resources resource_type
```

For information about a resource type, enter:

```
hatype -display [resource_type]
```

If *resource_type* is not specified, information regarding all types is displayed.

Resource Type Agents

For a list of agents whose values match given conditional statement, type:

```
haagent -list [conditional]
```

Conditional statements can take three forms:

```
Attribute=Value  
Attribute!=Value  
Attribute=~Value
```

Multiple conditional statements may be used and imply **AND** logic. If no conditional statement is specified, all agents in the cluster are listed.



For an agent's run-time information, type:

```
haagent -display [agent_name]
```

If *agent_name* is not specified, information regarding all agents is displayed.

Status Indicator	Definition
Faults	Indicates the number of agent faults and the time the faults began.
Messages	Displays various messages regarding agent status.
Running	Indicates the agent is operating.
Started	Indicates the file is executed by the VCS engine.

Clusters

For the value of a specific cluster attribute, type:

```
haclus -value attribute
```

For information about the cluster, type:

```
haclus -display
```

To modify a cluster attribute, type:

```
haclus -[help [-modify]]
```

To enable link monitoring, type:

```
haclus -enable LinkMonitoring
```

To disable link monitoring, type:

```
haclus -disable LinkMonitoring
```


Status Queries

For the status of all service groups in the cluster, including resources, type:

```
hastatus
```

For the status of a particular service group, including its resources, enter:

```
hastatus -group service_group [ -group service_group ]...
```

To display the current status in tabular format of all systems and a specific group and its resources (or all service groups if no group is specified, type:

```
hastatus [ -sound ] [ -group service_group ]
```

The `-sound` option causes a bell to ring each time a resource faults.

For the current status of faults in the VCS cluster, including faulted service groups, resources, systems, links, or agents, type:

```
hastatus -summary
```

Note Unless executed with the `-summary` option, `hastatus` continues to produce output that reflects online state transitions until you interrupt it with the keystroke command CTRL+C.



Basic Operations

System Operations

Forcing a System to Start

To force a system to start, type:

```
hasys -force system
```

This command overwrites the configuration on all systems in the cluster. Before using it, verify that the current VCS configuration is valid.

Modifying System Attributes

To modify a system's attributes, type:

```
hasys -modify modify_options
```

Some attributes are internal to VCS and cannot be modified. For details on system attributes, see "[The -modify Syntax](#)" on page 63.

Changing a System's Load Attribute

To change the load attribute of a specific system, type:

```
hasys -load system value
```

The load value is used for load balancing on failover. For details on load attributes, see the `hasys (1M)` manual page.

Displaying System Nodeid Value

To display the value of a system's nodeid as defined in the file `/etc/llttab`, type:

```
hasys -nodeid nodeid
```

Freezing a System

To freeze a system (disable groups and resources from being brought online or taken offline on the system), type:

```
hasys -freeze [-persistent] [-evacuate] system
```

The option `-persistent` enables the freeze to be remembered when the cluster is rebooted.

The option `-evacuate` fails over the system's active service groups to another system in the cluster before the freeze is enabled.



Thawing a System

To thaw or “unfreeze” a frozen system (reenable groups and resources to be brought online or taken offline on the system), type:

```
hasys -unfreeze [-persistent] system
```

Service Group Operations

Onlining a Group

To start a service group and bring its resources online, type:

```
hagrp -online service_group -sys system
```

Offlining a Group

To stop a service group and take its resources offline, type:

```
hagrp -offline service_group -sys system
```

Switching a Group

To switch a service group from one system to another, type:

```
hagrp -switch service_group -to system
```

The `-switch` option is valid for failover groups only. A service group can be switched only if it is online or partially online.

Freezing a Group

To freeze a service group (disable onlining and offlining), type:

```
hagrp -freeze service_group [-persistent]
```

The option `-persistent` enables the freeze to be remembered when the cluster is rebooted.

Thawing a Group

To thaw a service group (reenable onlining and offlining), type:

```
hagrp -unfreeze service_group [-persistent]
```



Enabling a Group

To enable a service group, type:

```
hagrp -enable service_group [-sys system]
```

A group can be brought online only if it is enabled.

Disabling a Group

To disable a service group, type:

```
hagrp -disable service_group [-sys system]
```

A group cannot be brought online or taken offline if it is disabled.

Enabling Group Resources

To enable all the resources in a service group, type:

```
hagrp -enableresources service_group
```

Resources can be brought online only if they are enabled.

Disabling Group Resources

To disable all the resources in a service group, type:

```
hagrp -disableresources service_group
```

Resources cannot be brought online or taken offline if they are disabled.

Flushing a Group

To flush a service group and enable corrective action, type:

```
hagrp -flush service_group -sys system
```

Caution The `-flush` option removes transitional resource states. As the states clear, a resource may be brought online inadvertently between monitor intervals. If a resource is brought online on another system during this period, data could be corrupted.

The `-flush` option halts the propagation of resource onlining or offlining through the resource configuration graph. For example, it removes transitional resource states (Istate) such as `Waiting to Go Online` and `Waiting for Children Online`. It also stops the propagation of offline operations. For example, it converts `Waiting to Go Offline/Propagate` to `Waiting to Go Offline`. However, it will not cancel an offline operation for an individual resource. If an offline operation is hung or has failed, you must take the resource offline manually.

Resource Operations

Onlining a Resource

To online a resource, type:

```
hares -online resource -sys system
```

Offlining a Resource

To offline a resource, type:

```
hares -offline resource -sys system
```

To offline a resource and propagate the command to its children, type:

```
hares -offprop resource -sys system
```

Similar to the service group `-offline` command, this command signals that its children should be taken offline. This action continues to the leaves of the resource's subtree.



Probing a Resource

To cause a resource's agent to immediately monitor the resource on a particular system, type:

```
hares -probe resource -sys system
```

Though the command may return immediately, the monitoring process may not be completed by the time the command returns.

Clearing a Resource

Clearing a resource automatically initiates the online process that was previously blocked while waiting for the resource to become clear.

To clear a faulted resource, initiate a state change from RESOURCE_FAULTED to RESOURCE_OFFLINE:

```
hares -clear resource [-sys system]
```

If *system* is not specified, the fault is cleared on all systems in the SystemList of the resource service group. (See also the service group command to clear faulted, non-persistent resources, `hagrps -clear`, on page 54.)

This command clears the resource's ancestors automatically. Persistent resources whose static attribute Operations is defined as None cannot be cleared with this command and must be physically attended to, such as replacing a raw disk. The agent then updates the status automatically.

Agent Operations

Under normal conditions, VCS agents are started and stopped automatically. To stop and start agents manually, type:

```
haagent -start agent_name -sys system  
haagent -stop agent_name -sys system
```

Reconfiguring VCS

In the following sections, each command is presented within the context of its function, for example, adding a system, service group, etc., and includes a description of the command's constraints. The relationship between the commands and the configuration language is also explained, including a description of how this language is used to modify a configuration.

Low-level commands permanently affect the configuration of the cluster. If the cluster is brought down gracefully with the command `hastop -all` or made read-only, the `main.cf` file and other configuration files written to disk reflect the updates.

A template for defining the values of various attribute types is also included in this chapter on page 75.

Note When specifying values in a command-line syntax, you must prefix values beginning with a dash (-) with a percentage sign (%). If a value begins with a percentage sign, you must prefix it with another percentage sign. (The initial percentage sign is stripped by the VCS engine and does not appear in the configuration file.)

Configuration Read/Write and Read-Only Modes

VCS must be in read/write mode before you can change the configuration. See page 50 for instructions.

The -modify Syntax

Most configuration changes are made using the `-modify` options of the commands `haclus`, `hagrp`, `hares`, `hasnmp`, `hasys`, and `hatype`. Specifically, the `-modify` option of these commands changes the attribute values stored in the VCS configuration file. By default, all attributes are global, meaning that the value of the attribute is the same for all systems.

Localizing Attributes

Non-static attributes of resource types may be localized, which enables you to specify a separate value for different systems. These attributes are localized on a per-resource basis. For example, to localize the attribute `attribute_name` for `resource` only, type:

```
hares -local resource attribute_name
```



Localizing the attribute means that for each system there is a value assigned in the service group's SystemList attribute. The original value is the same for each system, and it is the same as the original global value; however, values may be modified at the system level, as described in the following table.

Dimension	Task and Command
scalar	Replace a value: -modify [object] attribute_name value [-sys system]
vector	Replace list of values: -modify [object] attribute_name value [-sys system]
	Add list of values to existing list: -modify [object] attribute_name -add value [-sys system]
	Update list with user-supplied values: -modify [object] attribute_name -update entry_value ... [-sys system]
	Delete all values in list (you cannot delete an individual element of a vector): -modify [object] attribute_name -delete -keys [-sys system]
keylist	Replace list of keys (duplicate keys not allowed): -modify [object] attribute_name value ... [-sys system]
	Add keys to list (duplicate keys not allowed): -modify [object] attribute_name -add value ... [-sys system]
	Delete user-supplied keys from list: -modify [object] attribute_name -delete -keys ... [-sys system]
	Delete all keys from list: -modify [object] attribute_name -delete -keys ... [-sys system]

Dimension	Task and Command
association	Replace list of key-value pairs (duplicate keys not allowed): -modify [object] attribute_name value ... [-sys system]
	Add user-supplied list of key-value pairs to existing list (duplicate keys not allowed): -modify [object] attribute_name -add value ... [-sys system]
	Replace value of each key with user-supplied value: modify [object] attribute_name -update key value ... [-sys system]
	Delete a key-value pair identified by user-supplied key: modify [object] attribute_name -delete key ... [-sys system]
	Delete all key-value pairs from association: modify [object] attribute_name -delete keys [-sys system] Note If multiple values are specified and if one is invalid, VCS returns an error for the invalid value, but continues to process the others. In the following example, if sysb is part of the attribute SystemList, but sysa is not, sysb is deleted and an error message is sent to the log regarding sysa. hagrp -modify group1 SystemList -delete sysa sysb [-sys system]



Adding, Modifying, and Deleting Resource Types

▼ To add a resource type

```
hatype -add resource_type
```

After creating a resource type, use the command `haattr` to add its attributes. See page 67 for instructions.

Note By default, resource type information is stored in the `types.cf` configuration file. Follow the instructions below to include additional or modified resource types in the `main.cf` configuration file without shutting down VCS. Type:

```
hatype -modify resource_type SourceFile "./resource_type.cf"
```

The information regarding `resource_type` is stored in the file `config/resource_type.cf`, and a line `include resource_type.cf` is added to the `main.cf` file.

▼ To set the value of static attributes

```
hatype -modify ...
```

See “[The -modify Syntax](#)” on page 63 for details.

▼ To delete a resource type

```
hatype -delete resource_type
```

You must delete all resources of the type before deleting the resource type.

Adding, Modifying, and Deleting Attributes

Using the `haattr` command, you can define new attributes, change the default values, and delete attributes associated with resource types.

▼ To add a resource attribute to the VCS configuration

```
haattr -add resource_type attribute_name [ value_type ] [ dimension ] [ default ... ]
```

The variable `value_type` is a `-string` (default) or `-integer`.

The variable `dimension` is `-scalar` (default), `-keylist`, `-association`, or `-vector`.

The variable `default` is the default value of the attribute and must be compatible with the `value_type` and `dimension`. Note that this may include more than one item, as indicated by ellipses (`...`).

▼ To add static attributes

```
haattr -add -static resource_type attribute_name [ value_type ]  
[ dimension ] [ default ... ]
```

▼ To change the default value of a resource attribute

```
haattr -default resource_type attribute_name new_value ...
```

The variable `new_value` refers to the attribute's new default value.

▼ To delete a resource attribute

```
haattr -delete resource_type attribute_name
```

▼ To delete a static attribute

```
haattr -delete -static resource_type attribute_name
```



Adding Service Groups

▼ To add a service group to your cluster

```
hagrp -add service_group
```

The variable *service_group* must be unique among all service groups defined in the cluster.

This command initializes a service group that is ready to contain various resources. To employ the group properly, you must populate its `SystemList` attribute to define the systems on which the group may be brought online and taken offline. (A system list is an association of names and integers that represent priority values.)

▼ To modify a service group attribute

```
hagrp -modify service_group attribute_name value
```

The variable *value* represents:

```
system_name1 priority1 system_name2 priority2
```

During a failover, the VCS engine employs load-balancing to determine the optimal target machine to which faulted applications will fail over. However, if it cannot determine the best location, VCS selects the system with the highest priority, which equals the lowest number designated in the `SystemList` association. Populating the system list is a way to give “hints” to the VCS engine regarding which machine in a balanced cluster is best equipped to handle a failover.

In cases where multiple groups fail over simultaneously, VCS automatically distributes the service groups to various systems in the cluster according to a “round-robin” algorithm that begins with the system with the highest priority.

For example, to populate the system list of service group `groupx` with systems `sysa` and `sysb`, type:

```
hagrp -modify groupx SystemList sysa 1 sysb 2
```

Similarly, to populate the `AutoStartList` attribute of a service group, type:

```
hagrp -modify groupx AutoStartList sysa sysb
```

The `AutoStartList` attribute is a keylist that contains a list of all systems upon which a designated service group is automatically brought online at system boot. (The order of the list is irrelevant because VCS attempts to bring a service group online as soon as a system in the list is brought online.)

Note If set to 0, the `AutoStart` attribute (integer-scalar) ignores the `AutoStartList` attribute at system boot. If set to 0, the `PrintTree` attribute (integer-scalar) won't write the dependency tree to the configuration file when the VCS engine is stopped.

You may also define a service group as parallel. To set the Parallel attribute to 1, type the following command. (Note that the default for this attribute is 0, which designates the service group as a failover group.):

```
hagrp -modify groupx Parallel 1
```

This attribute cannot be modified if resources have already been added to the service group. See the template on page 75 for more information on defining attribute types.

Constraints

You cannot modify attributes created by the system, such as the state of the service group. You can modify the attributes SystemList, AutoStartList, and Parallel only by using the command `hagrp -modify`.



Modifying Service Group System Lists

If you are modifying a service group from the command line, the VCS server immediately updates the configuration of the group's resources accordingly.

For example, suppose you originally defined the SystemList of service group groupx as sysa, sysb. Then, after the cluster was brought up, you added a new system to the list:

```
hagrp -modify groupx SystemList -add sysc 3
```

The SystemList for groupx changes to sysa, sysb, sysc, and an entry for sysc is created in the group's resource attributes that are stored on a per-system basis. These attributes include information regarding the state of the resource on a particular system.

Next, suppose you made the following modification:

```
hagrp -modify groupx SystemList sysa 1 sysc 3 sysd 4
```

Using the option `-modify` without other options erases the existing data and replaces it with new data. Therefore, after making the change above, the new SystemList becomes sysa=1, sysc=3, sysd=4. The system sysb is deleted from the system list, and each entry for sysb in local attributes is removed.

When you have completed modifying the SystemList, entries for the local attributes of each resource in the service group are updated automatically.

Constraints

You can modify the populated SystemList only with the commands `-modify`, `-add`, `-update`, `-delete`, or `-delete -keys`.

If you modify the system list dynamically using the command `hagrp -modify` without other options (such as `-add` or `-update`), the service groups must first be taken offline on the systems being removed. The modification will fail if any service group is not offline completely.

If you modify the system list using the command `hagrp -modify` with the options `delete` or `-delete -keys`, any system to be deleted that is not offline will not be removed, but modifying the offline systems will proceed normally.

If you modify the system list to add a system that has not been defined by the command `hasys -add`, the system will not be added, but modifying other valid systems will proceed normally.



Adding Resources

▼ To add a resource

```
hares -add resource resource_type service_group
```

This command creates a new resource, *resource*, which must be a unique name throughout the cluster, regardless of where it resides physically or in which service group it is placed. The resource type is *resource_type*, which must be defined in the configuration language. The resource belongs to the group *service_group*.

When new resources are created, all non-static attributes of the resource's type, plus their default values, are copied to the new resource. Three attributes are also created by the system and added to the resource:

- ◆ Critical (default = 1). If the resource or any of its descendants faults while online, the entire service group is marked "faulted" and failover occurs.
- ◆ AutoStart (default = 1). If the resource is set to AutoStart, it is brought online in response to a service group command. All resources designated as AutoStart=1 must be online for the service group to be considered online. (This attribute is unrelated to AutoStart attributes for service groups.)
- ◆ Enabled. If the resource is set to Enabled, the agent for the resource's type manages the resource. The default is 1 for resources defined in the configuration file *main.cf*. The default is 0 for resources added on the command line. (Adding resources on the command line requires several steps, and the agent must be prevented from managing the resource until the steps are completed. For resources defined in the configuration file, the steps are completed before the agent is started.)

▼ To modify the attributes of the new resource

```
hares -modify resource attribute_name value
```

The variable *value* depends on the type of attribute being created.

To set a new resource's Enabled attribute to 1, type:

```
hares -modify resourceA Enabled 1
```

Constraints

Resource names must be unique throughout the cluster.

You cannot modify resource attributes defined by the system, such as the resource state.

The resource's agent is started on a system when its Enabled attribute is set to 1 on that system. Specifically, the VCS engine begins to monitor the resource for faults. Agent monitoring is disabled if the Enabled attribute is reset to 0.

Linking Resources

▼ To specify a dependency relationship between two resources

```
hares -link parent_resource child_resource
```

The variable *parent_resource* depends on *child_resource* being online before going online itself. Conversely, *parent_resource* must take itself offline before *child_resource* goes offline.

For example, before an IP address can be configured, its associated NIC must be available, so for resources IP1 of type IP and NIC1 of type NIC, specify the dependency as:

```
hares -link IP1 NIC1
```

Constraints

When linking resources, the parent cannot be a resource whose Operations attribute is equal to None or OnOnly. Specifically, these are resources that cannot be brought online or taken offline by an agent (None), or can only be brought online by an agent (OnOnly).

Loop cycles are automatically prohibited by the VCS engine. You cannot specify a resource link between resources of different service groups.

A resource can have an unlimited number of parents and children. For example, in the UNIX sample configuration on page 30, the resource *nfs_export1* of type Share cannot be brought online until *export1* and the Mount and NFS resources are brought online. Because resources of type NIC can only be brought online, and resources of type Disk cannot be brought online or taken offline, *groupx_hme0* and *c1t1d0s3* are “leaves” in this dependency graph.



Deleting and Unlinking Service Groups, and Resources

▼ **To delete a service group**

```
hagrp -delete service_group
```

▼ **To delete a resource**

```
hares -delete resource
```

Note that deleting a resource won't take the object being monitored by the resource offline. The object remains online, outside of the control and monitoring of VCS.

▼ **To remove the dependency between two service groups**

```
hagrp -unlink parent_group child_group
```

▼ **To remove the dependency relationship between two resources**

```
hares -unlink parent_resource child_resource
```

Note You can unlink service groups and resources at any time.

Constraints

You cannot delete a service group until all of its resources are deleted.

VCS High-Level Configuration Language and Low-Level Commands

The cluster configuration defined in the files `main.cf` and `types.cf` is loaded into the first system that boots in the cluster. The configuration's high-level language constructs are then converted into low-level commands that VCS understands. By mastering both the constructs and commands, you can use the high-level language to set up a solid initial configuration, and use low-level commands to make minor changes dynamically.

The following sample configuration on UNIX illustrates the correspondence between the language constructs and commands. Plain-style text represents high-level configuration language code from a sample `main.cf` file. Bold text represents the corresponding low-level commands.

Sample Configuration on UNIX

```
include "types.cf"
    // this will include all of the types information

cluster vcs
    haclus -modify ClusterName vcs

system sysa
    hasys -add sysa

system sysb
    hasys -add sysb

snmp vcs
    hasnmp -modify SnmpName vcs

group groupx (
    hagrp -add groupx

    SystemList = { sysa, sysb }
        hagrp -modify groupx SystemList sysa 0 sysb 1

    AutoStartList = { sysa }
        hagrp -modify groupx AutoStartList sysa

    )

Disk c1t1d0s3 (
    hares -add c1t1d0s3 Disk groupx

    Partition = { c1t1d0s3 }
        hares -modify c1t1d0s3 Partition c1t1d0s3

    )
```



```
IP nfssrvx (  
    hares -add nfssrvx IP groupx  
    Device = hme0  
        hares -modify nfssrvx Device hme0  
    Address = "192.2.40.11"  
        hares -modify nfssrvx Address "192.2.40.11"  
    )  
Mount export1 (  
    hares -add export1 Mount groupx  
    MountPoint = "/export1"  
        hares -modify export1 MountPoint "/export1"  
    BlockDevice = "/dev/sharedg/voloracle"  
        hares -modify export1 BlockDevice  
            "/dev/sharedg/voloracle"  
    Type = vxfs  
        hares -modify export1 FSType ufs  
    MountOpt = rw  
        hares -modify export1 MountOpt rw  
    )  
NFS NFS_groupx_16 (  
    hares -add NFS_groupx_16 NFS groupx  
    Nservers = 16  
        hares -modify NFS_groupx_16 Nservers 16  
    )  
NIC groupx_hme0 (  
    hares -add groupx_hme0 NIC groupx  
    Device = hme0  
        hares -modify groupx_hme0 Device hme0  
    Type = ether  
        hares -modify groupx_hme0 NetworkType ether  
    )
```



The VCS graphical user interface comprises the Cluster Manager and the Configuration Editor. Cluster Manager enables you to manage, configure, and administer the cluster while VCS is running (online). Configuration Editor is a configuration tool that enables you to generate new configuration files (`main.cf` and `types.cf`) while VCS is offline. Cluster Manager is described in the following sections. For more information on Configuration Editor, see page 127.

Note The VCS GUI is installed in a separate package, `VRTScscm`. See the accompanying installation guide for details.

Before Using Cluster Manager

Before you can use Cluster Manager, you must:

- ✓ Set the GUI display (page 78).
- ✓ Verify that the configuration has a user account. If a user account does not exist, you must create one. For more information on user management, see “[About Cluster Manager Users](#)” on page 78.

Note that a user account is established during the VCS installation procedure with the user name *admin* and the password *password*. This account provides you immediate access to Cluster Manager; however, to ensure proper security, we recommend that you modify the password.

- ✓ Start Cluster Manager (page 80).
- ✓ Create a cluster panel (page 81).



Setting the Display

Note Setting the display is not required on Windows NT workstations. The UNIX version of Cluster Manager requires an X-Windows desktop.

▼ To set the display on a UNIX workstation

1. Type the following command to grant the system permission to display on the desktop:

```
# xhost +
```

2. Configure the shell environment variable `DISPLAY` on the system where Cluster Manager will be launched. For example, if using Korn shell, type the following command to display on the system “myws”:

```
# export DISPLAY=myws:0
```

About Cluster Manager Users

There are three categories of users for Cluster Manager, based on the privileges assigned to them: Guest, Operator, and superuser/administrator.

- ◆ If the user account is set up with Guest privileges, the user can view and monitor the cluster. They cannot modify the configuration or perform administrative tasks. To create a user account with Guest privileges, prefix the user name with `VCSGuest` (uppercase, lowercase, or mixed case) when creating the user profile. For example, `VCSGUESTuser_name`, `vcsguestuser_name`, and `VCSGuestuser_name` are valid VCS user accounts with Guest privileges.
- ◆ If the user account is set up with Operator privileges (`VCSOpuser_name`), the user can view the cluster and perform basic administrative tasks, such as bringing service groups online and taking them offline. They cannot modify the configuration. To create a user account with Operator privileges, prefix the user name with `VCSOp` (uppercase, lowercase, or mixed case) when creating the user profile. For example, `VCSOpuser_name`, `vcsoptuser_name`, and `VCSOpuser_name` are valid VCS user accounts with Operator privileges.
- ◆ If the user account is set up as superuser or administrator, the user has full permission to view the cluster, perform administrative tasks, and modify the configuration. ***All user names that are not prefixed with `VCSGuest` or `VCSOp` are assigned Administrator privileges.***

Note VCS user names are case-sensitive. For example, if user Jan’s account is established as `VCSGUESTJAN` (all uppercase), she cannot access her account from user name `VCSGuestjan`. VCS treats both as different users.

Adding A User Account from the Command Prompt

To establish a user account for Cluster Manager, execute the following commands as root (UNIX) or Administrator (Windows NT) on any system in the cluster.

▼ To establish an account in the VCSGuest category

1. Set the VCS configuration file to read/write mode:

```
haconf -makerw
```

2. Type the following command to add the user:

```
hauser -add VCSGuestuser_name
```

3. Enter a password when prompted.

4. Confirm your password.

5. Set the VCS configuration file to read-only:

```
haconf -dump -makero
```

▼ To establish an account in VCSOperator category

1. Set the VCS configuration file to read/write mode:

```
haconf -makerw
```

2. Type the following command to add the user:

```
hauser -add VCSOpuser_name
```

3. Enter a password when prompted.

4. Confirm your password.

5. Set the VCS configuration file to read-only:

```
haconf -dump -makero
```



▼ **To establish an account in the superuser category**

1. Set the VCS configuration file to read/write mode:

```
haconf -makerw
```

2. Type the following command to add the user:

```
hauser -add user_name
```

3. Enter a password when prompted.

4. Confirm your password.

5. Set the VCS configuration file to read-only:

```
haconf -dump -makero
```

Starting Cluster Manager

- ◆ **On UNIX**, after establishing a user account and setting the display, type the following command to start Cluster Manager:

```
# hagui
```

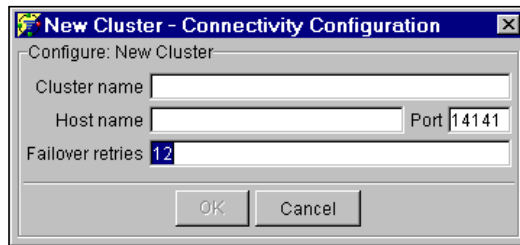
- ◆ **On Windows NT**, click the VERITAS Cluster Manager icon on the desktop.

Adding and Configuring a Cluster Panel

To administer a cluster using Cluster Manager, you must add a cluster panel or reconfigure an existing cluster panel in the Cluster Monitor, as instructed below. Note that to activate the connection of either procedure you must log on to the cluster after completing the final step. (See “[Logging On to and Off of a Cluster](#)” on page 99).

▼ To add a new cluster panel

1. In the Cluster Monitor, click File.
2. From the pull-down menu, click New cluster.



3. Enter a cluster name, hostname, and port number. (The default in the Failover Retries field is 12, and in the Port field is 14141.) The cluster name is the name of the cluster designated by the user. It is not necessarily the name of the cluster recognized by the engine. It appears on the local display only to differentiate the clusters being monitored.
4. Click OK. An inactive panel opens in the Cluster Monitor window.

▼ To configure a cluster panel

If the cluster is currently being monitored (active):

1. In the Cluster Monitor, right-click an active cluster panel.
2. From the pop-up menu, click Log out.
3. Right-click the inactive panel.
4. From the pop-up menu, click Configure.
5. Enter a hostname and a port number. (The default in the Failover Retries field is 12.)
6. Click OK.



If the cluster is not being monitored (inactive):

1. In the Cluster Monitor, right-click an inactive cluster panel.
2. From the pop-up menu, click Configure.
3. Enter a hostname and a port number. (The default in the Failover Retries field is 12.)
4. Click OK.

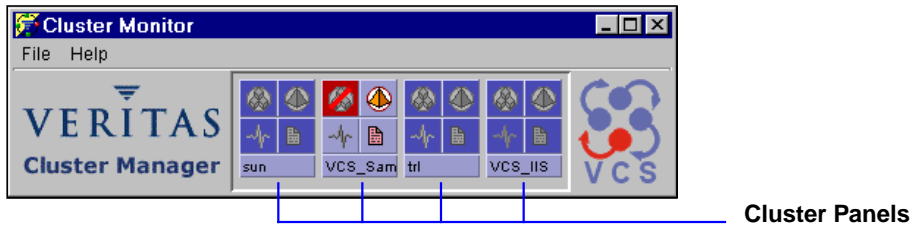
The Cluster Manager Windows

The main windows in Cluster Manager are:

- ◆ Cluster Monitor, from which you log on to and off of a cluster, view summary information regarding various cluster objects, customize the display, and exit Cluster Manager. For more information about the Cluster Monitor, see page 83.
- ◆ Cluster Explorer, from which you view the status of cluster objects and perform various operations. For more information about Cluster Explorer, see page 84.
- ◆ Command Center, from which you can build and execute VCS commands. For more information about Command Center, see page 92.
- ◆ Command Shell, from which you can launch non-interactive shell commands on cluster systems and view results on a per-system basis. For more information about the Command Shell, see page 92.
- ◆ Template View, from which you can view the resource dependencies and attributes defined in each template. You may also use the templates to add service groups to the cluster, or to copy the resources within the templates to existing service groups. For more information about the Templates View, see page 96.

The Cluster Monitor

After starting the GUI, the first window that appears is the Cluster Monitor window. It includes one or more panels that represent the clusters being monitored. The name of the cluster appears at the bottom of each panel.



The Cluster Monitor window includes pull-down menus, and each panel includes its own pop-up menus and pop-up windows. (To activate a pop-up menu, click the right mouse button. To activate a pop-up window, move the cursor over any object.) Information within each window changes depending on whether you are logged on to the cluster.

Each panel contains four icons representing service groups, member systems, heartbeats, and log messages.



A cluster panel is inactive (offline) and appears dimmed until the user logs in to the cluster. The panel then becomes active (online), and its background and icons change from gray to color.

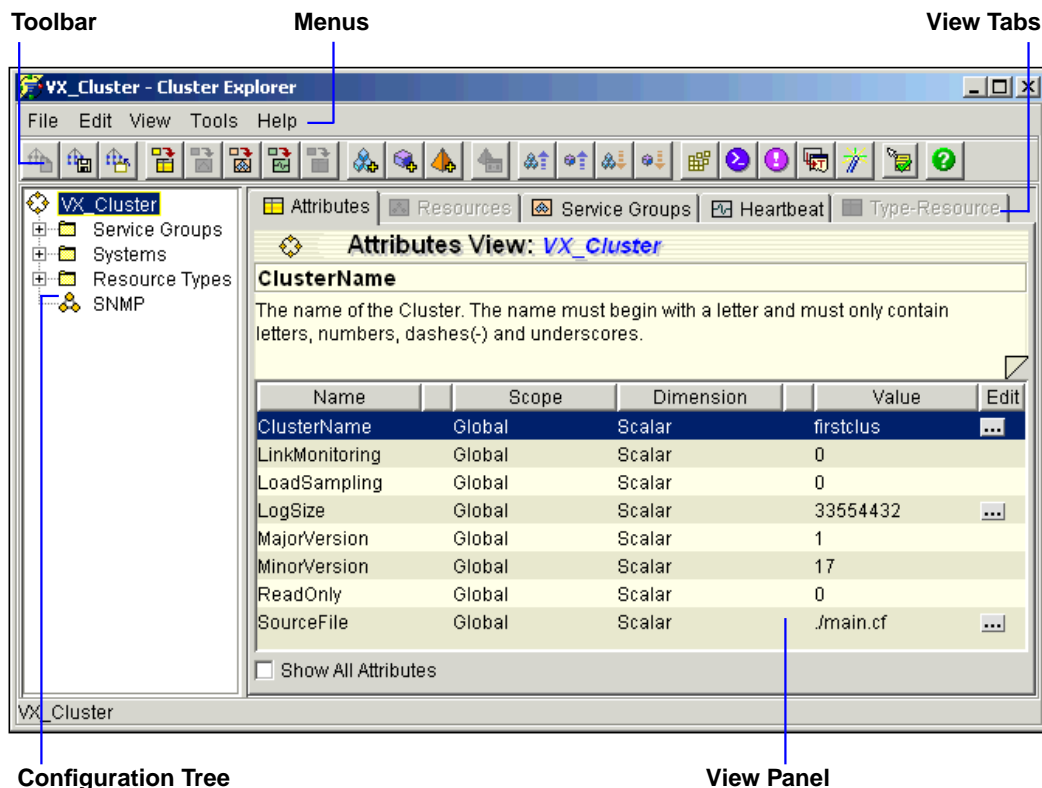
To administer a cluster, you must add a cluster panel or reconfigure an existing cluster panel in the Cluster Monitor. See page 99 for more information.

You can also change the font, color and sound preferences from this window. For more information on setting user preferences, see page 99.



Cluster Explorer

The Cluster Explorer window is the main window for cluster administration. It is divided into three sections, or “panes.”



The top pane includes a toolbar, which enables you to perform frequently used operations quickly. The Cluster Explorer toolbar is described beginning on page 91.

The left pane contains the configuration tree, a dynamic hierarchical display of all cluster objects.

The right pane contains a view panel, which displays details of the object selected in the configuration tree. The right pane also includes five tabs from which you can monitor and administer various cluster objects, including attributes, resources, service groups, heartbeats, and SNMP. Click any of the Cluster Explorer tabs to access the tab's view panels. View panels provide detailed information about the cluster configurations in tabular format or graphs. The Cluster Explorer view panels are described beginning on page 86.

To open the Cluster Explorer window, double-click the Service Groups, Member Systems, or Heartbeats icon in the Cluster Monitor panel, or right-click anywhere in the panel and select Explorer View from the drop-down menu.

The Cluster Explorer Toolbar

The Cluster Explorer toolbar contains 23 icons, each of which performs a specific operation.



From left to right:



Open a configuration. When you open a configuration, the configuration becomes read/write. This enables you to modify the configuration.



Save a configuration. When you save a configuration, the configuration is written to disk.



Close a configuration. When you close a configuration, the configuration is written to disk and becomes read-only.



Tear-off view for monitoring attributes. Click this icon to display the Attributes View window.



Tear-off view for monitoring resource dependencies. Click this icon to display the Resource View window.



Tear-off view for monitoring service group dependencies. Click this icon to display the Service Group Dependency View window.



Tear-off view for monitoring heartbeats. Click this icon to display the Heartbeat View window.



Tear-off view for monitoring type-resource dependencies. Click this icon to display the Type Resource View window.



Add a service group. Click this icon to display the Add Group dialog box.



Add a resource. Click this icon to display the Add Resource dialog box.





Add a system. Click this icon to display the Add System dialog box.



Manage all systems in the cluster. Click this icon to display the System Manager dialog box.



Bring a service group online. Click this icon to display the Online Group dialog box.



Bring a resource online. Click this icon to display the Online Resource dialog box:



Take a service group offline. Click this icon to display the Offline Group dialog box.



Take a resource offline. Click this icon to display the Offline Resource dialog box.



Launch Command Center. The Command Center enables you to build and execute VCS commands.



Launch the Cluster Shell. The Cluster Shell enables you to launch a shell command on cluster systems, and to view the results on a per-system basis.



Launch the Log Desk. The Log Desk displays messages received from the VCS engine and the commands issued from the GUI.



Display templates. Click this icon to display the Templates view.



Launch the service group configuration wizard. This wizard assists you in creating VCS service groups. From this window you can add a new service group to the cluster.



Show or hide tool tips. A red check mark indicates that they will be hidden.



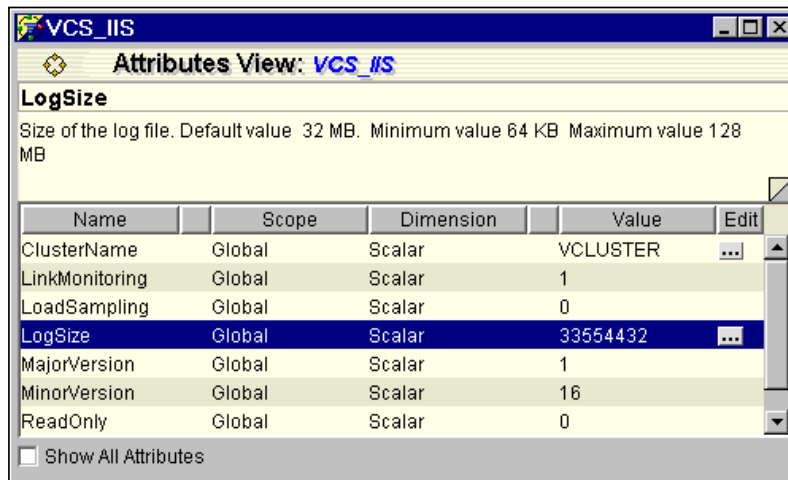
Access online Help from Cluster Manager or Configuration Editor. From this window you can navigate to specific task procedures, including modifying cluster objects, linking and unlinking service groups, issuing commands, etc.

Cluster Explorer View Panels

Cluster Explorer includes five view panels that provide detailed information about the cluster configurations in tabular format or graphs. Each view is described below.

Attributes View

The Attributes View panel displays the attributes of a selected cluster object in a tabular format.



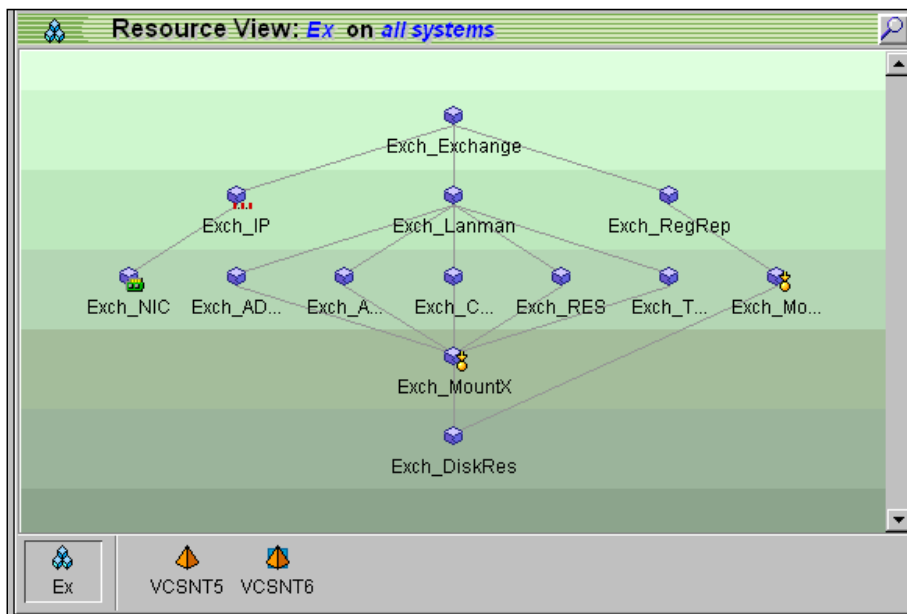
In the view panel, the attributes of the cluster object selected in the configuration tree are displayed in a tabular format consisting of five columns: Name, Scope, Dimension, Value, and Edit. You can expand the Scope column to view systems whose attributes are of local scope. You can also expand the Value row to display dimensions other than scalar. To expand or collapse columns and rows, click the triangular bullet. The Edit column on the right of the view panel denotes attributes that can be modified. For more information on how to edit attributes, see page 106.

For a complete list of VCS attributes, including their type, dimension, scope, and descriptions, see [Appendix C](#).



Resource View

The Resource View panel enables you to monitor resource dependencies.



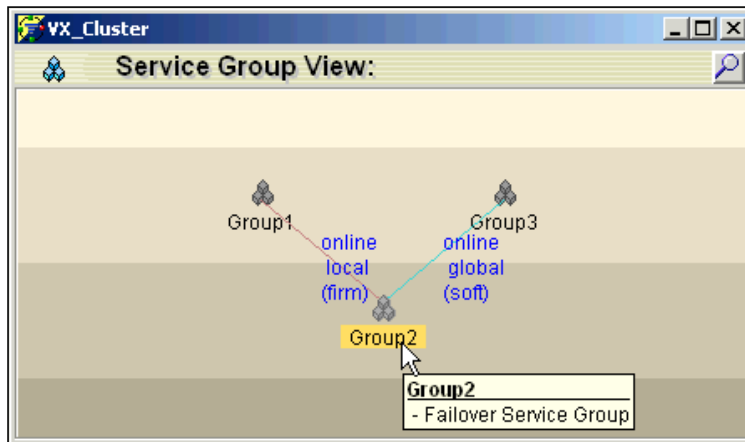
A dependency graph is displayed showing the resource dependencies within the service group. Resources are arranged according to their distance from the root resources. Root resources (resources without parents) are displayed in the top row. The line between the two resources represents a dependency, or “parent-child” relationship. (To review information on the VCS dependency structure, see “[Resource Dependencies](#)” on page 8.)

The graphic overlay and color scheme for each resource icon represent the state of the resource. The state can be the same for the resource across all systems (global), or can be the state of a resource on a specified system (local). A highlighted resource displays the direct links to and from the resources, and displays the resources (parents or children) at the opposite end of the link. The tool tip for each resource provides additional information, such as the resource type, its current state, and the values of other important attributes.

The bottom pane is the Group/System panel. This pane displays all systems running a particular service group if a service group is selected in the configuration tree.

Service Group View

The Service Group View panel enables you to monitor service groups and to view dependencies among them.

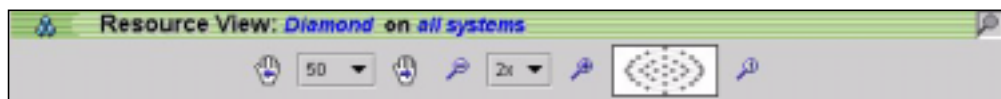


A dependency graph is displayed showing the group dependencies within the cluster. This graph is similar to that of the Resource View window, except in this view each icon represents a service group instead of a resource.

The graphic overlay and color scheme for each service group icon represent the state of the service group. The color of the link indicates the type of the dependency. For example, a blue link indicates a soft dependency and a red link indicates a firm dependency.

The Zoom Tool for Resource and Service Group Views

The Resource and Service Group view panels include a tool to zoom in or out of the graph view. This tool resembles a magnifying glass, and is located in the top-right corner. To view large configurations, click on the magnifying glass icon to open the zoom panel.



- ◆ To move the view to the left or right, select a number from the box between the hand icons and click the icon that represents the direction you want to move the view. (The number indicates the distance in screen pixels.)

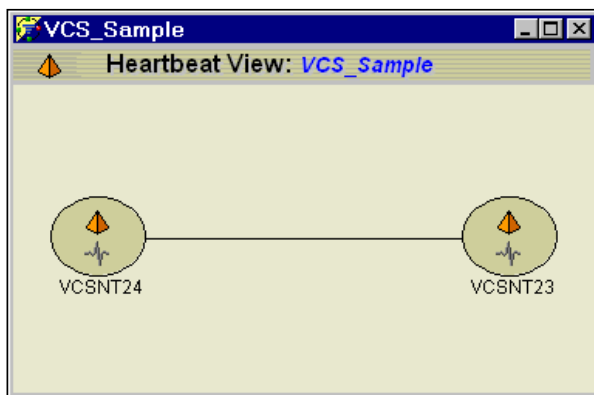


- ◆ To shrink or enlarge the view, select a size factor from the box between the magnifying glass icons and click the icon that represents how you want to modify the size of the view.
- ◆ To return to the original view, click the magnifying glass icon labeled 1.

To view a segment of the graph, place the cursor in the box to the right of the magnifying glass icon labeled +. A red box will appear that enables you to select a segment for viewing.

The Heartbeat View

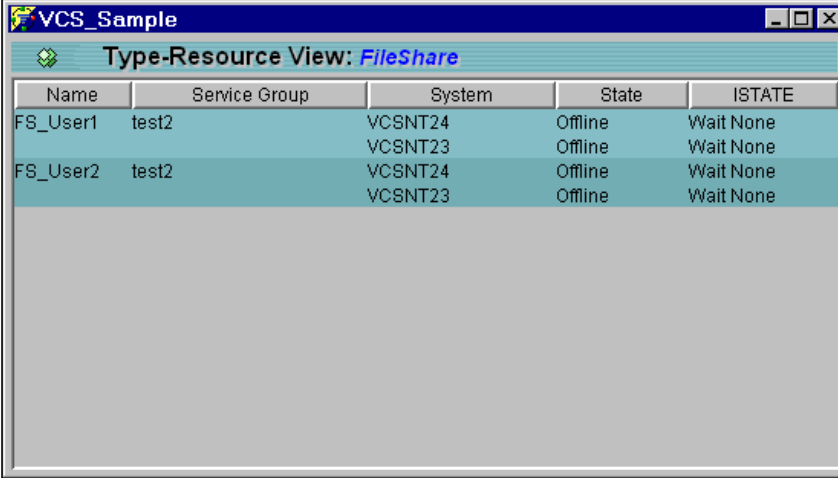
The Heartbeat View panel enables you to monitor heartbeats.



In the view panel, the system heartbeat graph is displayed with systems arranged as nodes. If a system in the cluster is having difficulty connecting to other systems, it appears with an error icon indicating that the link or disk heartbeat is down. The system icon shows the system status, and the link icon shows the link status. Gray icons indicate that the link is not being monitored. The tool tip also displays details of the link and the status of the disk group heartbeat.

Type-Resource View

The Type-Resource View panel enables you to monitor resources of a selected resource type.



Name	Service Group	System	State	ISTATE
FS_User1	test2	VCSNT24	Offline	Wait None
		VCSNT23	Offline	Wait None
FS_User2	test2	VCSNT24	Offline	Wait None
		VCSNT23	Offline	Wait None

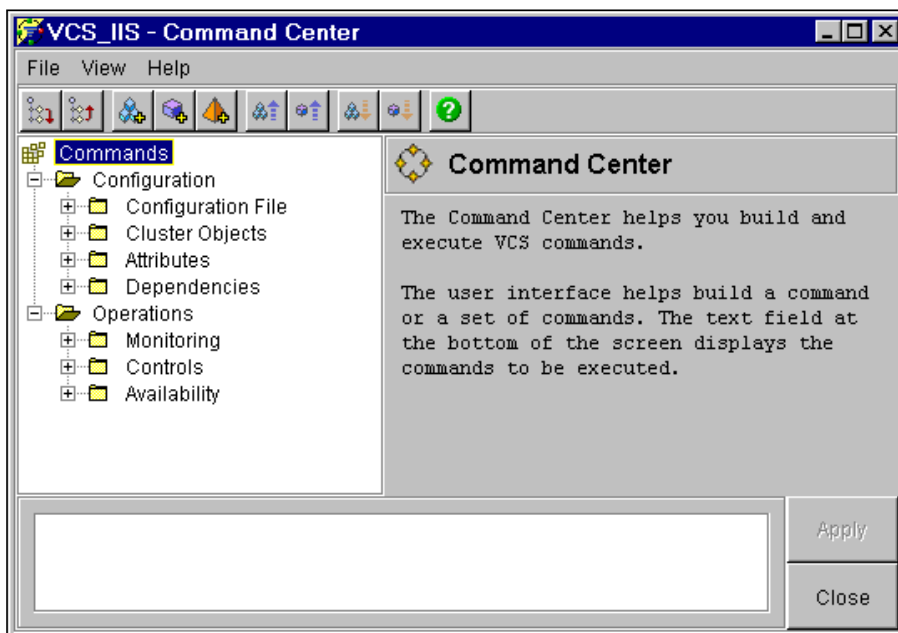
In the view panel, resources are presented in a tabular format consisting of five columns: Name, Service Group, System, State, and ISTATE (intermediate state). The columns display information for all systems on which the resource is configured. For example, click the State column to view resources displayed according to their state. Note that if you select a resource type from the Resource Types folder, the columns display information for resources belonging to the resource type across all service groups. If you select a resource type from the Service Groups folder, the columns display information only for resources belonging to the resource type within the selected service group.



The Command Center

Command Center enables you to build and execute VCS commands.

To launch Command Center, click the  icon in the Cluster Explorer toolbar. You may also launch this window by right-clicking a panel of the Cluster Monitor window and selecting it from the pop-up menu.



Command Center is divided into three panes: a command tree on the left that includes all VCS operations, and a view panel on the right that includes a description of the selected command. The bottom pane displays the commands being executed.

The command tree is organized according to the type of VCS operation. Beneath the top folder (Commands) there are two subfolders: Configuration and Operations. Each contains several subfolders, and each subfolder contains commands for specific operations. Click on the icon to the left of the Configuration or Operations folder to display their subfolders and commands. A list of operations supported by Command Center is provided in the following section.

Supported Commands

Command Tree Folder	Subfolder	Operations
Configuration	Configuration File	Open Configuration
		Save Configuration
		Close Configuration
	Cluster Objects	Add Service Group
Add Resource		
Add System		
Delete Service Group		
Delete Resource		
Delete System		
Attributes	Modify Cluster Attributes	
	Modify Service Group Attributes	
	Modify Resource Attributes	
	Modify System Attributes	
	Modify Resource Type Attributes	
Dependencies	Link Resources	
	Link Service Groups	
	Unlink Resources	
	Unlink Service Groups	
Operations	Monitoring	Enable Link Monitoring
		Disable Link Monitoring
	Controls	Online Service Group
		Online Resource
		Offline Service Group
		Offline Resource
		Switch Service Group
		Offprop Resource
	Availability	Probe Resource
		Freeze Service Group
		Freeze System



Command Tree Folder	Subfolder	Operations
Operations	Availability	Unfreeze Service Group Unfreeze System Enable Service Group Enable Resources for Service Group Disable Service Group Disable Resources for Service Group Autoenable Service Group Clear Resource Flush Service Group

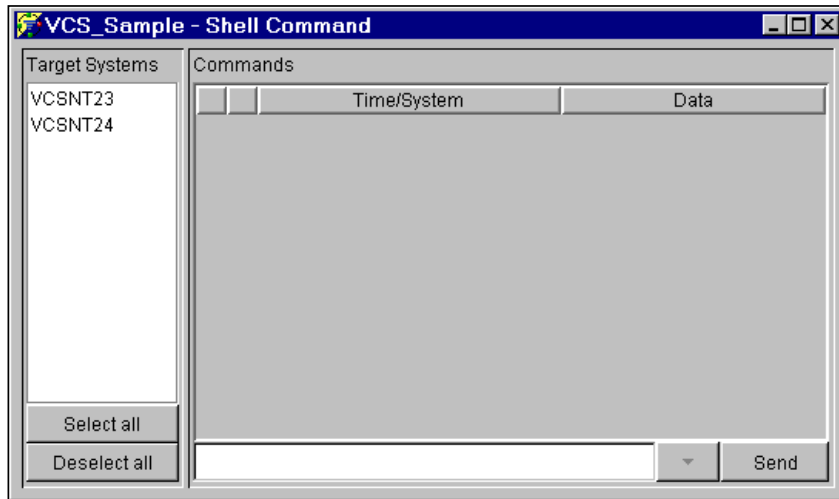
▼ To issue commands from Command Center

1. Select the command from the command tree. If necessary, expand the tree to view the command.
2. The right panel displays the interface for the selected command. If necessary, select cluster objects and additional options.
3. Click Apply.

The Cluster Shell

The Cluster Shell enables you to launch a non-interactive shell command on one or more cluster systems, and to view the results on a per-system basis. Note that commands issued from Cluster Shell carry the full privilege of root (UNIX) and Administrator (Windows NT). See “VCS Security” on page 166 for details.

To launch the Cluster Shell, click the  icon in the Cluster Explorer toolbar.



This tab is also divided into three panes. The left view panel displays a list of the cluster systems. The right view panel displays the command in a tabular format, including the time the command was issued and the system on which it was issued. Output from the command is also displayed as generated on the system’s command line or console. The bottom pane is an editable field in which you enter the command. To launch a command, click Send.

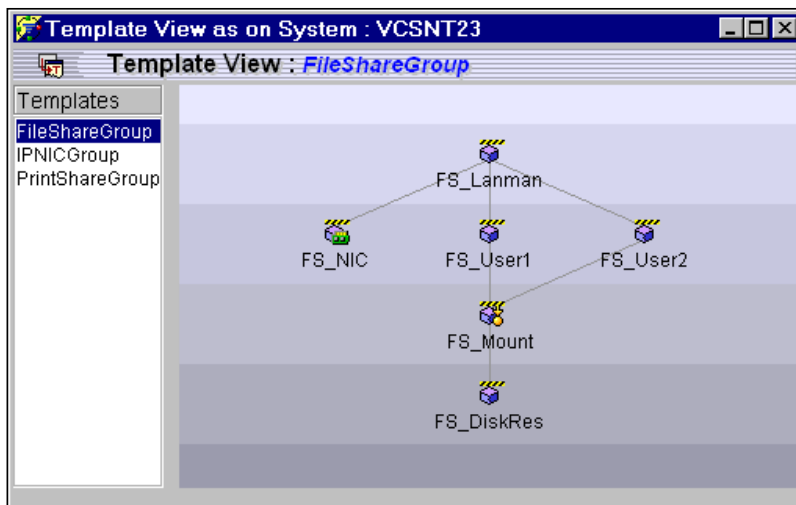
Note If arguments to a command entered from this window contain a space or a backslash (\), enclose the arguments within double quotation marks. For example: dir “c:\” or dir “c:\program files.”



The Template View Window

The Template View displays the service group templates available in VCS.

Click the  icon in the Cluster Explorer toolbar to display the Template View.



From this window you may view the resource dependencies and attributes included in each template. You may also use the templates to add service groups to the cluster, or to copy the resources within the templates to existing service groups.

This window is divided into two panes. The left pane contains the template list, which displays the templates available on the system to which Cluster Manager is connected. The right pane displays the selected template's resource dependency graphs.

About VCS Templates

Templates are predefined service groups and can be used to create service groups in your configuration file. A template conforms to the VCS configuration language, and contains the definition of service group, resources within the service group, resource attributes, and dependencies within resources.

A template file includes the extension `.tf`. Templates are located at the following paths:

- ◆ On **UNIX**: `etc\VRTSvcs\templates`
- ◆ On **Windows NT**: `%VCS_HOME%\templates`

Sample Template

```
group TestGroup (  
  )  
  FileOnOff File_1 (  
    PathName = file_path  
  )  
  
  FileOnOff File_2 (  
    PathName = file_path  
  )  
File_2 requires File_1
```



Managing the Cluster

Use Cluster Manager to perform the following operations:

- ◆ Add and configure a cluster panel (page 81).
- ◆ Log on to and off of a cluster (page 99).
- ◆ Manage users (page 99).
- ◆ Set user preferences (page 103).
- ◆ Get status information about the cluster and its objects (page 105).
- ◆ Open, save, and close a cluster configuration (page 105).
- ◆ Monitor cluster objects including attributes, resources, service groups, heartbeats, and resource types (page 106).
- ◆ Edit attributes of cluster objects (page 106).
- ◆ Add and delete service groups, resources, and systems (page 109).
- ◆ Manage dependencies between resources and between service groups (page 114).
- ◆ Manage systems for a service group (page 118).
- ◆ Bring service groups and resources online and take them offline (page 119).
- ◆ Create a new service group using the wizard (page 123).
- ◆ View log information (page 125).
- ◆ Import additional resource types (page 126).



Logging On to and Off of a Cluster

▼ To log on to a cluster

1. Click on the Cluster Monitor panel that represents the cluster you want to monitor. The Logon window opens on the desktop.
2. Enter a valid VCS user name and password.
3. Click OK. The animated display shows various cluster objects, such as service groups and resources, being transferred from the server to the GUI. (The Cluster Explorer window is launched automatically upon initial logon.)

If the logon was successful, the icons in the Cluster Monitor panel appear in color, indicating the various system states. If unsuccessful, repeat steps 1–3.

Note that the default configuration assumes that the VCS engine is running on the local system. If it is not, you must add a cluster panel or reconfigure an existing cluster panel. See page 99 for instructions.

▼ To log off of a cluster

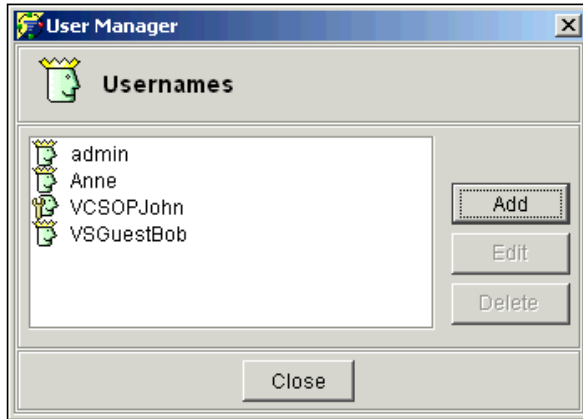
1. Right-click on an active Cluster Monitor panel.
2. From the pop-up menu, click Log Out. The panel becomes inactive, and its background and icons turn gray. All windows related to the cluster close, and you may be prompted to save the configuration if any commands were executed on the cluster.

Managing Users from Cluster Manager

The VCS User Manager enables you to add, edit, and delete user profiles. Open the User Manager from the Cluster Explorer's File menu (File>User Manager). For more information on Cluster Manager users and their privileges, see "[About Cluster Manager Users](#)" on page 78.



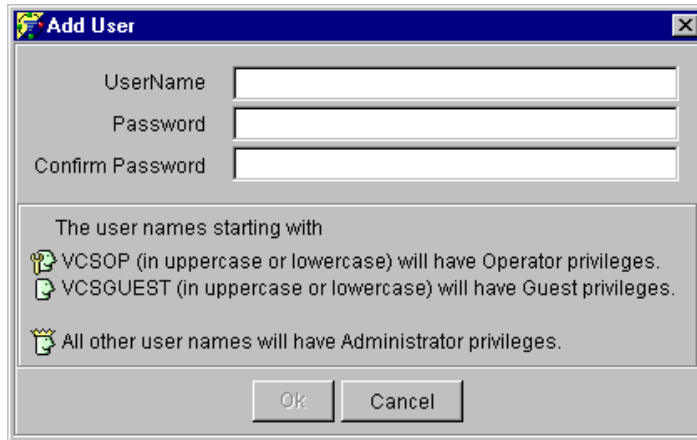
The icons next to the user name indicate the privileges associated with the user. In the following example, users admin and Anne have Administrator privileges, user John has Operator privileges, and user Bob has Guest privileges.



Note The User Manager feature is enabled only for users logged on with Administrator privileges. Users with Operator privileges can change their own password.

▼ To add a user

1. From Cluster Explorer, open the User Manager (File>User Manager).
2. Click Add.



3. In the Add User dialog box, enter the name of the user. To associate Guest privileges with the user, prefix the username with VCSGuest. To associate Operator privileges with the user, prefix the username with VCSOp. Note that these prefixes could be in uppercase, lowercase, or in mixed case. (For more information about VCS users, see [“About Cluster Manager Users”](#) on page 78.)
4. Enter the password.
5. Confirm the password.
6. Click OK.



▼ **To change a user password (Administrator)**

1. Open the Edit User dialog box.
 - a. From Cluster Explorer, open the User Manager (File>User Manager).



- b. Select the user whose password is to be changed.
 - c. Click Edit.
2. In the Edit User dialog box, enter the new password.
3. Reenter the password in the Confirm Password field.
4. Click OK.

▼ **To change a user password (Operator)**

1. From Cluster Explorer, select Change Password from the File menu.
2. In the Edit User dialog box, enter the new password.
3. Reenter the password in the Confirm Password field.
4. Click OK.

▼ **To delete a user**

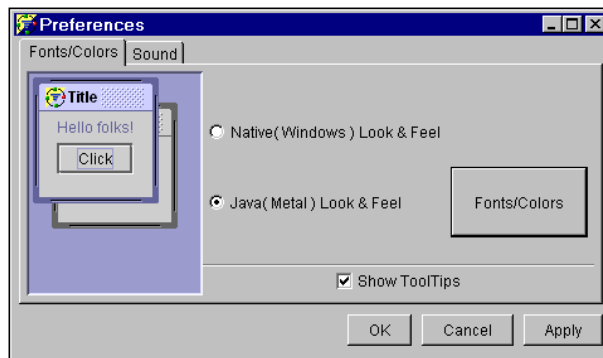
1. From Cluster Explorer, open the User Manager (File>User Manager).
2. Select the user to delete.
3. Click Delete.

Setting User Preferences

The Cluster Monitor enables you to set the visual display according to your preferences. It also enables you to disable the sound feature, or associate sound with specific events.

Font/Color Display

1. From the Cluster Monitor, open the Preferences dialog box (File>Preferences).



2. Select the Fonts/Colors tab. Choose one of the two options described below:

▼ **To select the Windows look and feel:**

- a. Select the Native (Windows) Look and Feel option.
- b. Click OK.

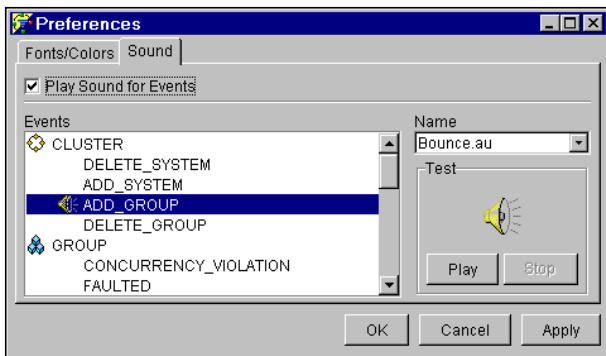
▼ **To select the Java look and feel:**

- a. Select the Java (Metal) Look and Feel option.
- b. Click Fonts/Colors to change font and color display.
- c. Select the desired color from the Color drop-down list.
- d. Select the desired font size from the Font Size drop-down list.
- e. Click OK to close the Fonts/Colors dialog box.
- f. Click OK to close the Preferences dialog box.



Sound Information

1. From the Cluster Monitor, open the Preferences dialog box (File>Preferences).
2. Select the Sound tab.



▼ To disable sound

1. Verify that the Play Sound for Events checkbox is not selected.
2. Click OK.

▼ To associate sounds with specific events

1. Verify that the Play Sound for Events checkbox is selected.
2. Select an event from the Events tree.
3. Select a sound from the Name drop-down list to associate with the selected event.
4. To test the selected sound, click Play.
5. Click Apply.
6. Repeat [step 1](#) through [step 5](#) to select events with which to associate sounds.
7. Click OK.




Getting Status Information on the Cluster and Its Objects

Cluster objects include systems, service groups, heartbeats and log messages.

For *summary* information on the cluster and its objects, use the Cluster Monitor pop-up windows and icons. For *detailed* information on the cluster and its objects, use the Cluster Explorer pop-up windows and icons.

- ◆ Pop-up windows enable you to view the status of a particular cluster object by moving the cursor to the object's icon. A pop-up window displaying the object's status will appear.
- ◆ Cluster icons indicate the states of various objects in the cluster by displaying the object in a specific color setting. For example:
 - ◆ In the Cluster Monitor window, a flashing red slash indicates that Cluster Manager's connection to the cluster has failed and that it will attempt to connect to another system. A flashing yellow slash indicates problems with the Cluster Manager's connection with the cluster.
 - ◆ In the Cluster Explorer window, a red slash indicates at least one object is faulted and no objects are online. A yellow slash indicates some objects are faulted and some objects are online. Additionally, an object that appears as gray or dimmed is offline, and a lock on the icon indicates that the object is frozen.

Opening, Saving, and Closing a Cluster Configuration

- ◆ To open a configuration, click the  icon in the Cluster Explorer toolbar. When you open a configuration, the configuration becomes read/write. This enables you to modify the configuration.
- ◆ To save a configuration, click the  icon in the Cluster Explorer toolbar. When you save a configuration, the configuration is written to disk.
- ◆ To close a configuration, click the  icon in the Cluster Explorer toolbar. When you close a configuration, the configuration is written to disk and becomes read-only.



Monitoring Cluster Objects

The Cluster Explorer view panel enables you to monitor cluster objects. It also enables you to create tear-off views to monitor cluster objects continuously.

Use Cluster Explorer to monitor the following:


- ◆ object attributes (page 106)
- ◆ resource dependencies (page 107)
- ◆ service groups (page 108)
- ◆ heartbeats (page 108)
- ◆ resource types (page 108)

Monitoring Object Attributes


The Attributes view enables you to monitor cluster object attributes, including the attributes of clusters, service groups, resources, and resource types. You may also edit attributes.

▼ To monitor attributes

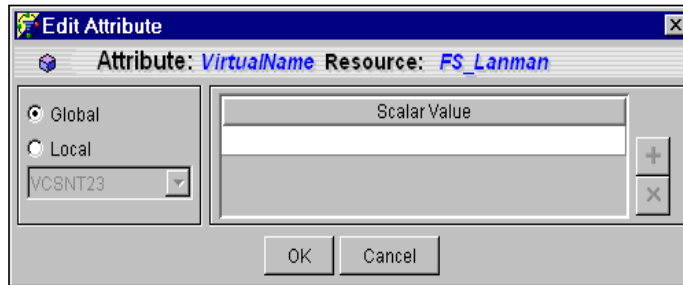
1. In the Cluster Explorer configuration tree, select the cluster object.
2. Select the Attributes tab in Cluster Explorer to open the Attributes View.

To create a tear-off of this view, click the  icon in the toolbar.

▼ To edit attributes

1. Open the Attributes View in one of three ways:
 - ◆ Click the Attributes tab in Cluster Explorer.
 - ◆ Click the  icon in the Cluster Explorer toolbar to create a tear-off.
 - ◆ In Command Center, select Modify *cluster_object* Attributes.
(Commands>Configuration>Attributes>Modify *cluster_object* Attributes)


- In the Attributes View window, click the small box labeled with ellipses (...). The Edit Attribute dialog box appears:



- Enter the changes to the attributes values. For non-scalar values, use the + and X buttons to add or delete elements. To change the attribute's scope, select the Global or Local options. For local attributes, select the system from the drop-down list.
- Click OK. (Click Apply if editing attributes from Command Center.)

Monitoring Resource Dependencies


The Resource View enables you to monitor resource dependencies. You may open the Resource View one of two ways:

- ◆ Click the Resources tab in Cluster Explorer.
- ◆ Click the  icon in the Cluster Explorer toolbar to tear-off the Resource View.




Monitoring Service Group Dependencies

The Service Group View enables you to monitor service groups. You may open the Service Group View one of two ways:

- ◆ Click the Service Groups tab in Cluster Explorer.
- ◆ Click the  icon in the Cluster Explorer toolbar to tear-off the Service Group View.


Monitoring Heartbeats

The Heartbeat View enables you to monitor heartbeats. You may open the Heartbeat View one of two ways:

- ◆ Click the Heartbeats tab in Cluster Explorer.
- ◆ Click the  icon in the Cluster Explorer toolbar to tear-off the Heartbeat View.


Monitoring Resource Types

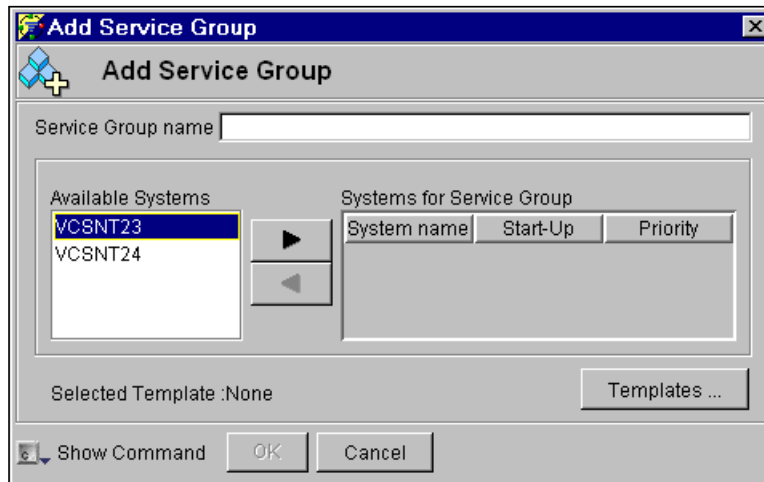
The Type-Resource View enables you to monitor resources of a selected resource type. You may open the Type-Resource View one of two ways:

- ◆ Click the Type-Resource tab in Cluster Explorer.
- ◆ Click the  icon in the Cluster Explorer toolbar to tear-off the Type-Resource View.

Adding and Deleting Service Groups

▼ To add a service group

1. Open the Add Service Group dialog box using one of the three methods described below:
 - ◆ Right-click Service Group or the cluster name in the Cluster Explorer configuration tree and select Add Service Group from the menu.
 - ◆ Click the  icon in the Cluster Explorer toolbar.
 - ◆ In Command Center, select Add Service Group (Commands>Configuration>Cluster Objects>Add Service Group).



2. In the Available Systems box, select the system on which the service group will be added.
3. Click the right arrow to move the available system to the Systems for the Service Group box.
4. Enter the name of the service group in the Service Group Name box.
5. Edit attributes according to your configuration.



6. To base the new service group on a template, click **Templates**. This launches the **Templates View** described on page 96. Otherwise, click **OK**. (Click **Apply** in **Command Center**.)

Note You may also add a service group from the **Configuration Wizard**. See page 123 for instructions.

▼ To delete a service group

From Cluster Explorer:


1. In the configuration tree, select service group to delete.
2. Right-click the service group name.
3. Select **Delete** from the drop-down menu.

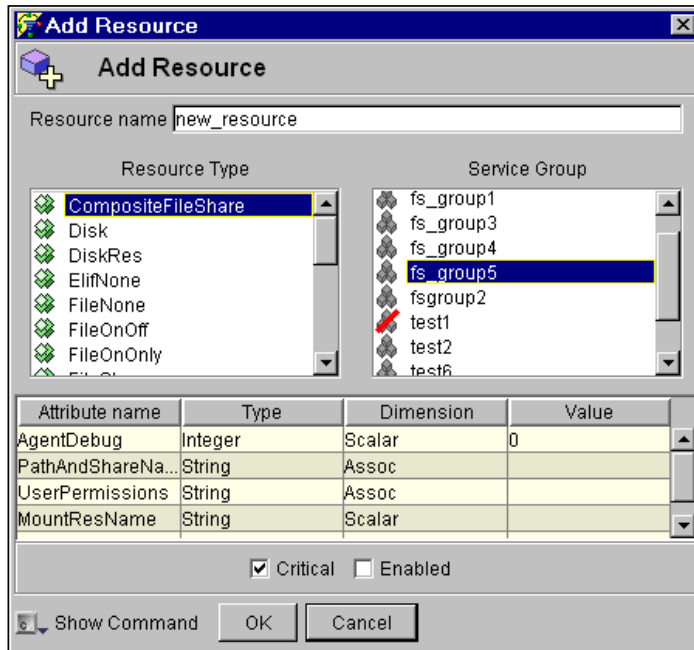
From Command Center:

1. Click **Delete Service Group** (**Commands**>**Configuration**>**Cluster Objects**>**Delete Service Group**).
2. Select the group to be deleted.
3. Click **Apply**.

Adding and Deleting Resources

▼ To add a resource

1. Open the Add Resource dialog box using one of the three methods described below:
 - ◆ Right-click Service Group or the cluster name in the Cluster Explorer configuration tree and select Add Resource from the menu.
 - ◆ Click the  icon in the Cluster Explorer toolbar.
 - ◆ In Command Center, select Add Resource (Commands>Configuration>Cluster Objects>Add Resource).



2. Enter the name of the resource in the Resource Name box.
3. Select the resource type from the Resource Type box.
4. Select the service group from the For Group box.
5. Click the checkbox to designate the resource as Critical, Enabled, or both.
6. Click OK. (Click Apply in Command Center.)



▼ **To delete a resource**

From Cluster Explorer:


1. In the configuration tree, select resource to delete.
2. Right-click the resource name.
3. Select Delete from the drop-down menu.

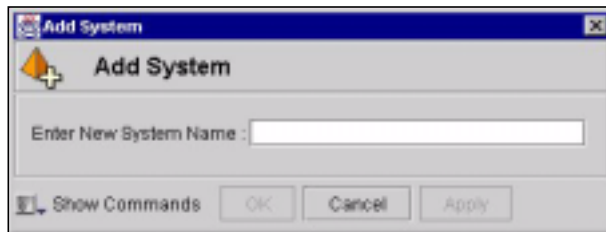
From Command Center:

1. Click Delete Resource (Commands>Configuration>Cluster Objects>Delete Resource).
2. Select the resource to be deleted.
3. Click Apply.

Adding and Deleting Systems

▼ To add a system

1. Open the Add System dialog box using one of the three methods described below:
 - ◆ Right-click System or the cluster name in the Cluster Explorer configuration tree and select Add System from the menu.
 - ◆ Click the  icon in the Cluster Explorer toolbar.
 - ◆ In Command Center, select Add System (Commands>Configuration>Cluster Objects>Add System).



2. Enter the name of the system to add.
3. Click OK. (Click Apply in Command Center.)

▼ To delete a system

1. In Command Center, click Delete System (Commands>Configuration>Cluster Objects>Delete System).
2. Select the system to be deleted.
3. Click Apply.



Managing Resource and Service Group Dependencies

▼ To link resources

From Cluster Explorer:

1. Click the Resources tab. A resource dependency graph is displayed.
2. To link a parent resource with a child:
 - a. Click on the first resource that is to be the parent.
 - b. Drag the yellow line to the resource that is to be the child and click.
3. Click Yes in the dialog box to confirm the connection.

Note If you decide to cancel the operation after performing step 2, click anywhere on the window's background. The yellow line is removed from the screen.

From Command Center:

1. Click Link Resources (Commands>Configuration>Dependencies>Link Resources).
2. From the list box, select the service group whose resources are to be linked.
3. Select the parent resource. After selecting the parent resource, the corresponding child resources are displayed in the Child Resources box.
4. Select a child resource.
5. Click Apply.

▼ **To unlink resources**

From Command Center:

1. Click Unlink Resources (Commands>Configuration>Dependencies>Unlink Resources).
2. From the list box, select the service group whose resources are to be unlinked.
3. Select the parent resource. After selecting the parent resource, the corresponding child resources are displayed in the Child Resources box.
4. Select a child resource.
5. Click Apply.

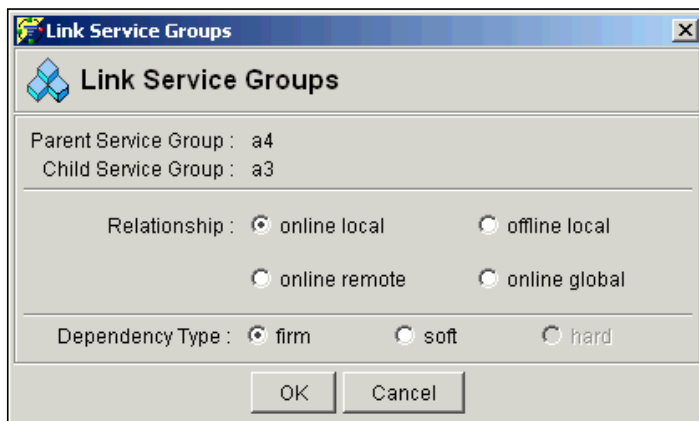
▼ **To link service groups**

From Cluster Explorer:

1. Click the Service Groups tab. A service group dependency graph is displayed.
2. To link a parent service group with a child:
 - a. Click on the first service group that is to be the parent.
 - b. Drag the yellow line to the resource that is to be the child and click.



3. In the Link Service Groups dialog box, click the corresponding option button to specify the relationship between the groups.



4. Click the corresponding option button to specify the dependency type. For more information on service group dependencies, see **section XXX on page XXX**.
5. Click OK.

From Command Center:

1. Click Link Service Groups (Commands>Configuration>Dependencies>Link Service Groups).
2. Select the parent resource. After selecting the parent group, the corresponding child service groups are displayed in the Child Service Groups box.
3. Select a child service group.
4. Specify the dependency from the Relationships option box.
5. Click Apply.

▼ To unlink service groups:**From Cluster Explorer:**

1. In the Service Group view, right-click the link to delete.
2. Click Yes when prompted to confirm the action.

From Command Center:

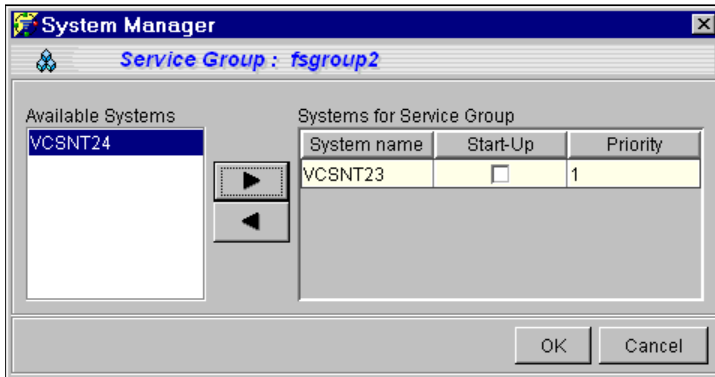
1. Click Unlink Service Groups (Commands>Configuration>Dependencies>Unlink Service Groups).
2. Select the parent resource. After selecting the parent group, the corresponding child resources are displayed in the Child Service Groups box.
3. Select a child service group.
4. Click Apply.



Managing Systems for a Service Group

▼ To modify a service group's system list from System Manager

1. Click the  icon in the Cluster Explorer toolbar to open System Manager.




2. Select the system from the Available Systems box.
3. Click the right arrow to move the available system to the Systems for Service Group box.
4. Click the checkbox in the Start Up column to designate if the system is to be started automatically.
5. Double-click the entry in the Priority column and enter the new value. The Priority value determines which system a service group fails over to if multiple system exist.
6. Click OK.

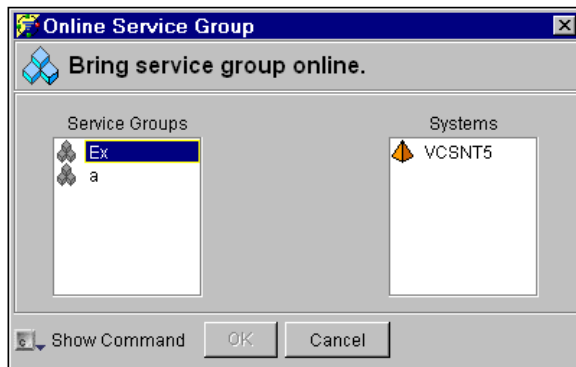
Bringing Service Groups Online

From Cluster Explorer:

1. Right-click the service group name or the service group icon.
2. Select to online the group on a cluster system (right-click>Online>*system*).

From the Online Service Group dialog box (Cluster Explorer toolbar):

1. Click the  icon in the Cluster Explorer toolbar to open the Online Service Group dialog box.



2. In the Service Groups box, select the service group to be brought online.
3. In the Systems box, select the system on which to bring the group online.
4. Click OK.

From Command Center:

1. Click Online Service Group (Commands>Operations>Controls>Online Service Group).
2. Select a service group.
3. Select a system on which to bring the group online.
4. Click Apply.



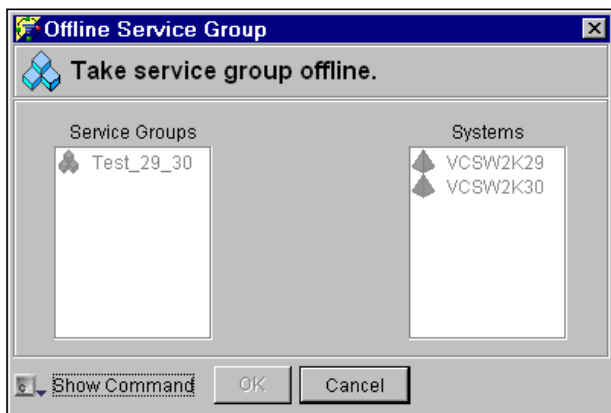
Taking Service Groups Offline

From Cluster Explorer:

1. Right-click the service group name or service group icon.
2. Select to offline the group on a cluster system (right-click>Offline>*system*).

From the Offline Service Group dialog box (Cluster Explorer toolbar):

1. Click the  icon in the Cluster Explorer toolbar to open the Offline Service Group dialog box.



2. In the Service Groups box, select the service group to be taken offline.
3. In the Systems box, select the system on which to take the service group offline.
4. Click OK.

From Command Center:


1. Click Offline Service Group (Commands>Operations>Controls>Offline Service Group).
2. Select a service group.
3. Select a system on which to take the service group offline.
4. Click Apply.

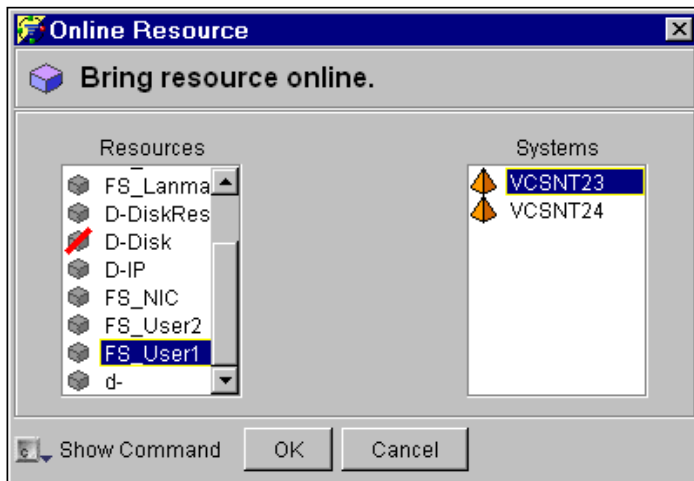
Bringing Resources Online

From Cluster Explorer:

1. Right-click the resource name.
2. Select to online the resource on a cluster system (right-click>Online>*system*).

From the Online Resource dialog box (Cluster Explorer toolbar):

1. Click the  icon in the Cluster Explorer toolbar to open the Online Resource dialog box.



2. In the Resources box, select the resource to be brought online.
3. In the Systems box, select the system on which to bring the resource online.
4. Click OK.



From Command Center:

1. Click Online Resource (Commands>Operations>Controls>Online Resource).
2. Select a resource.
3. Select a system on which to bring the resource online.
4. Click Apply.

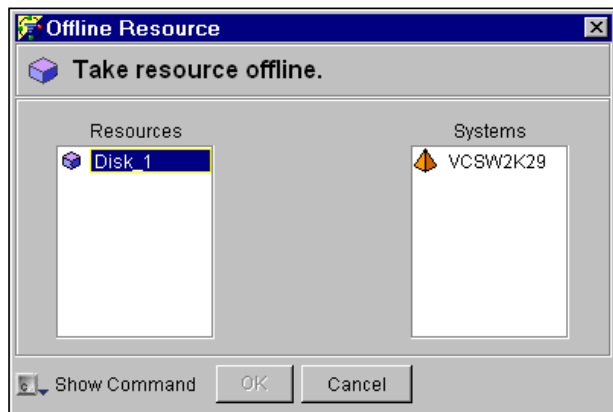
Taking Resources Offline

From Cluster Explorer:

1. Right-click the resource name.
2. Select to offline the resource on a cluster system (right-click>Offline>*system*).

From the Offline Resource dialog box (Cluster Explorer toolbar):

1. Click the  icon in the Cluster Explorer toolbar to open the Online Resource dialog box.




2. In the Groups box, select the resource to be taken offline.
3. In the Systems box, select the system on which to take the resource offline.
4. Click OK.

From Command Center:

1. Click Offline Resource (Commands>Operations>Controls>Offline Resource).
2. Select a resource.
3. Select a system on which to take the resource offline.
4. Click Apply.

Creating a New Service Group Using the Wizard

1. Click the  icon in the Cluster Explorer toolbar to launch the service group configuration wizard.
2. Select the name of the service group and the system on which it is to be configured (the target system):
 - a. Enter the name of the service group.
 - b. Select the target system.
 - c. Click the right arrow.
 - d. Indicate whether the system will start automatically and in which order the system will start (priority). A check mark in the box indicates that the system will start automatically. System priority is numbered sequentially, with 1 denoting that the system will start first following a failover.
 - e. Click Next.

Note To remove a system from the system list of the service group, select the system and click the left arrow.


3. Confirm whether or not you are basing the service group on a predefined template:
 - a. Select the template on which to base the new service group. If you are not using a template, click Finish to proceed with the configuration.
 - b. Click Next. If applicable, a window is displayed notifying you that the name of a resource within the new service group is already in use. This window also includes a list of alternative names. You may select an alternative name, or modify the name. Click Next after resolving the name clashes.

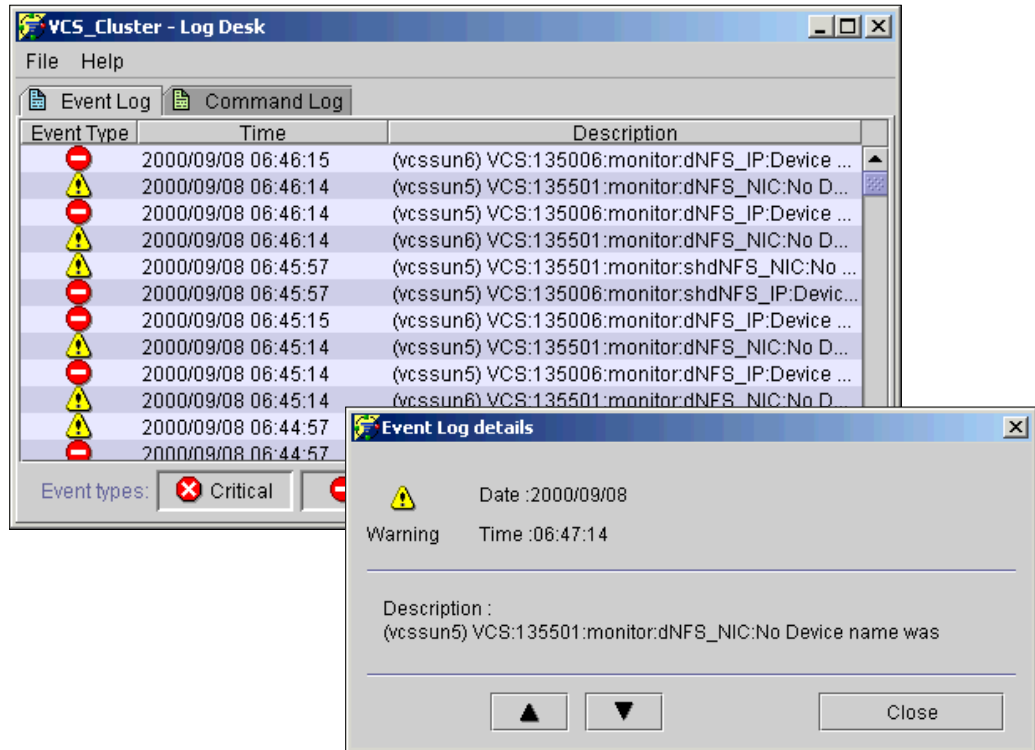


- c. The list box on the left displays the templates available on the system to which Cluster Manager is connected. The resource dependency graph of the templates, the number of resources, and the resource types are also displayed.
 - d. Click Next again after viewing the template. You are prompted to create the service group based on the selected template.
 - e. Click Next. A window is displayed indicating that the commands are being sent to add the group, its resources, and the attributes and dependencies specified in the template. A progress indicator displays the percentage of the commands fired. The actual commands are displayed at the top of the indicator.
 - f. Click Next when prompted that the service group has been successfully created.
- 4. Review the service group's resource attributes:**
- a. A window is displayed listing the service group's resources and their associated attributes. If necessary, you may modify the default values of the resources according to your specific configuration requirements. Click Finish to accept the default values and complete the configuration. Follow steps a through f below to modify the attributes.
 - b. Select the resource from the list on the left.
 - c. Select the attribute to be modified.
 - d. Click the ellipses (...) at the end of the table row.
 - e. In the Edit Attribute dialog box, enter the attribute values. To modify the scope of the attribute, click the option buttons for Global or Local. For non-scalar values, use the + and X buttons to add or delete items.
 - f. Click OK.
 - g. Repeat the procedure for each resource and attribute.
- 5. Click Finish.**

Viewing Log Information

The Log Desk enables you to monitor log messages received from the VCS engine, and view commands issued from Cluster Manager.

To launch the Log Desk, click the  icon in the Cluster Explorer toolbar. You may also launch this window by right-clicking a panel of the Cluster Monitor window and selecting it from the pop-up menu.



The Log Desk window includes two tabs: Log and Command. The Log tab displays in tabular format the event type, the time the event occurred, and a description of the messages. Log messages provide a history of the messages logged by the engine since the engine was started. The most recent message appears at the top of the list.

VCS event types are displayed as buttons at the bottom of the tab. These buttons indicate the category of the event. Click an Event Type button to select the types of messages to display or hide in the log table.

The Command tab displays in tabular format a log of commands issued by Cluster Manager to the cluster. This log includes the time the command was issued, its type, and its status (success or failure). The most recent command appears at the top of the list. Only commands issued in the current session are displayed.



To browse the Log Desk window for detailed views of each log message or command, double-click on the event's description. Use the arrow buttons in the Event Log Detail pop-up window to navigate backward and forward through the list. You may also launch this window by right-clicking a panel of the Cluster Monitor window and selecting it from the pop-up menu.

Importing Additional Resource Types Information

VCS allows you to import additional resource types into your configuration.

Note You cannot import resource types that already exist in your configuration.

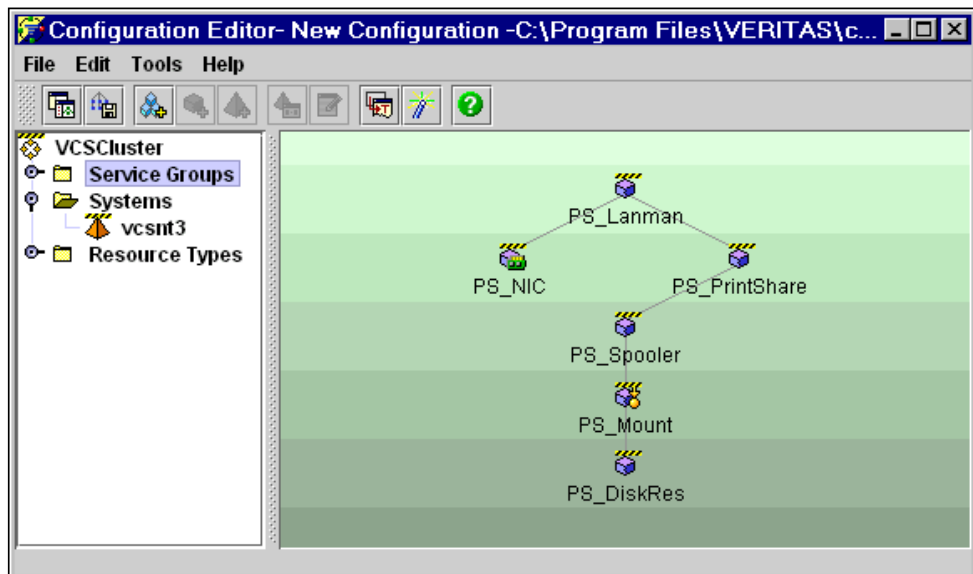
1. From the Cluster Explorer's File menu, select Import Types.
2. From the File Selection dialog box, select the file from which you want to import the resource type. Note that the dialog box displays files on the system that Cluster Manager is connected to, and not necessarily your local system.
3. Click Import. Various messages appear indicating the progress of the operation. The imported types appear in Cluster Explorer.

Configuration Editor

Configuration Editor is a tool that enables you to generate new configuration files (main.cf and types.cf) while VCS is offline. It also enables you to edit existing configuration files. It includes a set of predefined templates from which you can configure standard applications as service groups. Templates consist of service group resources, including their attributes and dependencies. Resource dependency graphs are provided for each template, and a configuration wizard is available to assist you in creating and modifying service groups. Configuration Editor also enables you to modify existing configuration files. For example, you can load the existing main.cf or types.cf configuration file, modify the file using the options available from Configuration Editor, and save the configuration.

Configuration Editor includes options for adding service groups, resources, and systems to your existing configuration. Additionally, you can specify system details for the service group, such as whether the group will start automatically on the system, and the system's failover priority. You can also save the configuration file on your system. To start VCS with the changed configuration, save the configuration file in the directory VCS_HOME/conf/config.

The Configuration Editor window is divided into three panes. The top pane contains the Configuration Editor toolbar. The left pane contains the configuration tree, a dynamic hierarchical display of the cluster, including its service groups, their resources and resource types. The right pane contains a view panel that displays the resource dependency graph of the selected service group or template. Double-click on the object to view or modify its attributes.



What is a Service Group Dependency?

A service group dependency provides a mechanism by which two service groups can be linked by a dependency rule, similar to the way resources are linked. Specifically:

- ◆ A service group that depends on other service groups is a *parent group*.
- ◆ A service group on which the other service groups depend is a *child group*.
- ◆ A service group can function as a parent and a child.

In service group dependencies, a link can be configured based on the following criteria:

- ◆ The category of the dependency, such as online or offline (described in “[Categories of Service Group Dependencies](#)” on page 131).
- ◆ The location of the dependency, such as local, global, or remote (described in “[Location of Dependency](#)” on page 132).
- ◆ The type of dependency, such as soft or firm (described in “[Type of Dependency](#)” on page 134).

Based on the type of link, VCS takes appropriate action to bring the parent/child service group online or take it offline, in response to group fault. The link also controls where VCS brings a group online during events such as a resource fault, automatic group start, system shutdown, etc.



Why Configure a Service Group Dependency?

While defining a cluster configuration, typically a service group and an application have a one-to-one relationship. For example, a service group hosts an application, or an application is contained within a service group. In a distributed computing environment there may be multiple applications running within a cluster, and one application may depend on another. For example, a database server may have several database applications depending on its services. In such situations, it is imperative that a dependency rule governing the onlining and offlining of service groups be specified.

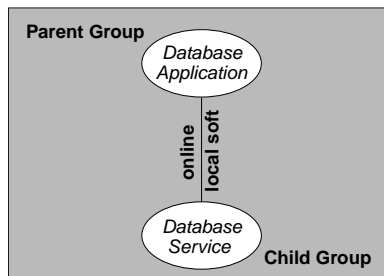
In our example, we can define a rule that requires a database server (a child group) to be online before any or all database applications (parent group) can be brought online. We can also define a rule that requires database applications to fail over when the database server faults. For example, database applications cannot be brought online until the database server is online. If the database server faults, the database applications cannot continue to provide services.

Note Adding service group dependencies adds complexity to your configuration. We strongly recommend evaluating various scenarios before implementing group dependencies in your environment. In general, an application should not be divided into multiple groups. Multiple applications correspond to multiple groups and group dependencies can be used to leverage failover scenarios.

Categories of Service Group Dependencies

Online Group Dependency

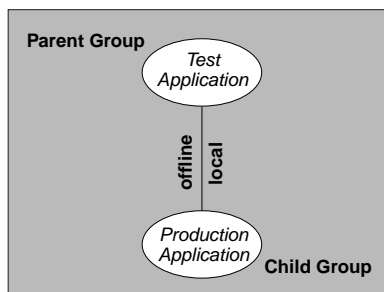
In an *online group dependency*, the parent group must wait for the child group to be brought online before it can start. For example, to configure a database application and a database service as two separate groups, you would specify the database application as the parent, and the database service as the child. The following illustration shows an *online local soft dependency* (described in “[Soft Dependency](#)” on page 134).



Online Local Soft Dependency Between a Parent Group and a Child Group

Offline Group Dependency

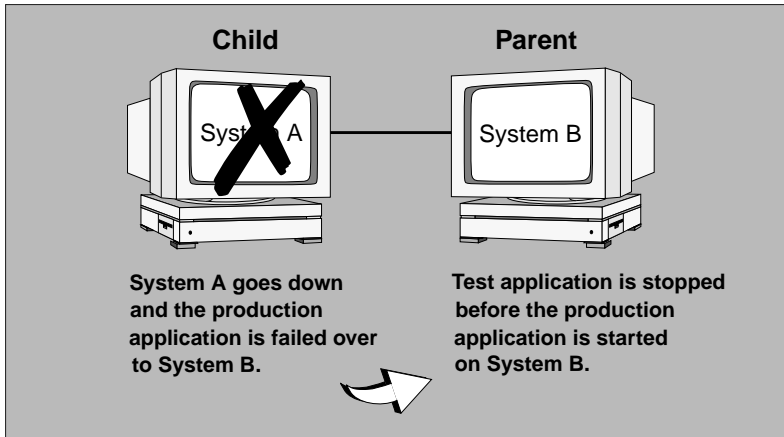
In an *offline group dependency*, the parent group can be started only if the child group is offline on the system, and vice versa. This prevents conflicting applications from running on the same system. For example, to configure a production application on one system and a test application on another, the test application must be the parent, and the production application must be the child. The following illustration shows an *offline local dependency*.



Offline Local Dependency Between a Parent Group and a Child Group



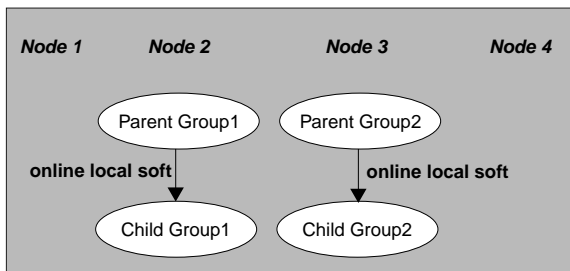
As illustrated below, System A failed while running the production application, causing the application to fail over to System B. Before VCS can restart the production application on System B, it must first stop the system's test application.



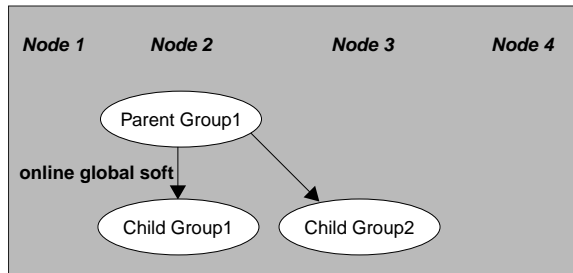
Location of Dependency

The location of the dependency determines the relative location of parent and child groups. Note that in the following examples, parent and child groups can be failover or parallel.

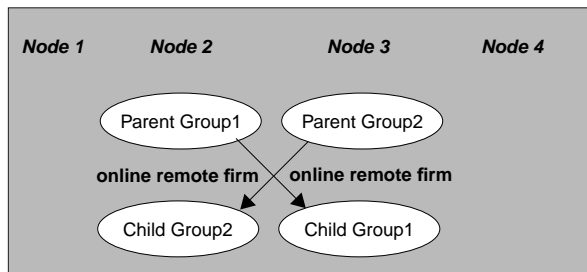
In a *local* dependency an instance of the parent group depends on an instance of child group being online on the same system. In the following figure, the failover parent group 1 depends on failover child group 1 online local soft. Failover parent group 2 depends on failover child group 2 online local soft.



In a *global* dependency an instance of parent group depends on an instance of the child group being online on any system. In the following figure, failover parent group 1 depend on parallel child groups online global soft. Child groups 1 and 2 are instances of the child group.



In a *remote* dependency an instance of parent group depends on an instance of the child group being online on any system other than the system on which the parent is online. In the following figure, the failover parent group 1 depends on the failover child group 1 online remote firm. The failover parent group 2 depends on the failover child group 2 online remote firm.



Type of Dependency

The type of dependency defines the rigidity of the link between parent and child groups. There are two types: *soft* and *firm*.

Soft Dependency

Soft dependency means that VCS imposes minimal constraints while onlining parent/child groups. The only constraint is that to bring the parent group online, the child group must be online. In a soft dependency, the parent is not automatically taken offline when the child faults. The location of the link designates whether or not a parent group will fail over after a fault and failover of the child group. For example:

- ◆ A link configured as *online local soft* designates that when the child group faults and fails over, the parent group fails over to the same system on which the child group was brought online.
- ◆ A link configured as *online global soft* designates that when the child group faults and fails over, the parent group never fails over (remains where it is).
- ◆ A link configured as *online remote soft* designates that when a child group faults and child group selects as the target system the system on which the parent is online, the parent group is taken offline. When a child group fails over, the parent group is brought online on another system within the cluster.

For more information regarding dependency types, see “[Overview of Service Group Dependency Configurations](#)” on page 136.

Soft dependency provides the following enhanced flexibility:

- ◆ VCS does not immediately take the parent offline if the child group faults.
- ◆ When both groups are online the child group can be taken offline while the parent is online and vice versa (the parent group can be taken offline while the child is online).
- ◆ The parent remains online if the child group faults and cannot fail over.
- ◆ To link a parent and child group with soft dependency, it is not required that the child group be online if parent is online. However, if the child group is also online, the parent and child may not be linked in such a way that their online states conflict with the type of link between parent and child.

Firm Dependency

Firm dependency means that VCS is more strict when onlining parent/child groups. In addition to the requirement of a soft dependency, a firm dependency requires that the parent group be taken offline if the child group faults. When the child is brought online on another system, the parent group is brought online on any system other than the system on which it was taken offline. For example:

- ◆ A link configured as *online local firm* designates that the parent group is taken offline when the child group faults. When the child group fails over to another system, the parent group is then migrated to that system.
- ◆ A link configured as *online global firm* designates that the parent group is taken offline when the child group faults. When the child group fails over to another system, the parent group is migrated to another system.
- ◆ A link configured as *online remote firm* is similar to online global firm, with the exception that the parent group is brought online on any system other than the system on which the child group was brought online.

In addition to the constraints imposed by soft dependency, firm dependency also includes the following constraints:

- ◆ If the child group faults, VCS takes the parent group offline as well. When the child group is brought online, the parent is brought online.
- ◆ When both groups are online, the child group cannot be taken offline while the parent group is online. However, the parent group can be taken offline while the child is online.
- ◆ If child group faults, the parent is taken offline. If the child cannot failover, the parent remains offline.
- ◆ To link a parent and child group with firm dependency, the parent group must be offline or the parent and child group must be online in such a way that their online states do not conflict with the type of link between parent and child.

Both soft and firm dependencies allow that if the parent group faults, the child group doesn't. The parent group may or may not failover, depending on the link constraints (such as *online local* versus *online global*). For example, for a failover parent and failover child linked with online local soft/firm dependency, if the parent faults, it cannot failover to another system. However, if they were linked with an online global soft/firm dependency, and if the parent faults, it can.



Overview of Service Group Dependency Configurations

In the following sections the term “instance” applies to parallel groups only. If a parallel group is online on three systems, each instance of the group is considered to be online on one system. For failover groups, only one instance of a group is online at any time.

The default type cited in the tables below is assumed when only the category of the dependency (online/offline) and the location (local/global/remote) are specified.

Failover Parent/Failover Child

Dependency Configuration	Definition	Behavior
online local soft	Failover parent group soft depends on failover child group being online on the same system.	Parent can be brought online on a system, for example, SystemA, only if the child is online on SystemA. If the child faults, the parent is not taken offline. After the child successfully fails over to another system, for example, SystemB, VCS migrates the parent to SystemB. If the child cannot fail over, the parent remains online on SystemA.
online local firm (default)	Failover parent group firm depends on failover child group being online on the same system.	Parent can be brought online on a system, for example, SystemA, only if the child is online on SystemA. If the child faults, the parent is taken offline on SystemA. When a child successfully fails over to another system, for example SystemB, VCS migrates the parent to SystemB. If child cannot fail over, parent remains offline.
online global soft	Failover parent group soft depends on failover child group being online anywhere in the cluster.	Parent can be brought online as long as a child group is running somewhere in the cluster. Parent remains online when the child faults and fails over, or when the child faults and cannot fail over.



Dependency Configuration	Definition	Behavior
online global firm (default)	Failover parent group firm depends on failover child group being online anywhere in the cluster.	Parent can be brought online as long as a child group is running somewhere in the cluster. For example, the parent group is online on SystemA, and the child group is online on SystemB. When the child faults on SystemB, the parent group on SystemA is taken offline. When the child successfully fails over to another system, for example, SystemC, VCS brings the parent online on another system, for example, SystemB. If child group cannot fail over, parent group remains offline.
online remote soft	Failover parent group soft depends on failover child group being online on any other system in the cluster.	Parent can be brought online on any system other than the system on which the child is online. For example if child group is online on SystemB, the parent group can be online on SystemA. When the child faults on SystemB, the parent remains online on SystemA unless VCS selects SystemA as the target system on which to bring the child group online. In that case, the parent is taken offline. After the child successfully fails over to SystemA, VCS brings the parent online on another system, for example SystemB.
online remote firm (default)	Failover parent group firm depends on failover child group being online on any other system in the cluster.	Parent can be brought online on any system other than the system on which the child is online. For example if child group is online on SystemA, the parent group can be online on SystemB. The parent is taken offline on SystemB when the child faults on SystemA. After the child successfully fails over to SystemB, VCS brings the parent online on another system, for example, SystemC.



Dependency Configuration	Definition	Behavior
offline local	Failover parent group depends on failover child group being offline on the same system and vice versa.	<p>Parent can be brought online on any system as long as the child is not online on the system, and vice versa.</p> <p>For example, if child group is online on System B, the parent can be brought online on System A. If the child faults on System B, and if VCS selects System A as the target on which to bring the child online, the parent on System A is taken offline and the parent is brought online.</p>

Failover Parent/Parallel Child

Dependency Configuration	Definition	Behavior
online local soft	Failover parent group soft depends on an instance of the child group being online on the same system.	Failover group can be brought online on any system, for example System A, only if an instance of the child group is online on System A. If an instance of the child group on System A faults, the parent cannot migrate until the child has successfully failed over. After the child fails over to another system, for example, System B, the parent migrates to System B. If the child cannot fail over, because no suitable system was found, or because the number of online instances of the child group equals the number of systems on which the child group can run, the parent may continue to run System A.
online local firm (default)	Failover parent group firm depends on an instance of the child group being online on the same system.	Failover group can be brought online on any system, for example, System A, only if an instance of the child group is online on System A. If the instance of the child group on System A faults, the parent is taken offline. After the child has successfully failed over to another system, for example System B, the parent then fails over to System B.

Dependency Configuration	Definition	Behavior
online global soft	Failover parent group soft depends on all instances of the child group being online anywhere in the cluster.	Failover group can be brought online anywhere as long as an instance of the child group is online somewhere in the cluster. Parent does not fault if an instance, or all instances, of the child group fault.
online global firm (default)	Failover parent group firm depends on all instances of the child group being online anywhere in the cluster.	Failover group can be brought online anywhere as long as an instance of the child group is online somewhere in the cluster. For example, if two instances of the child are online on System A and System B, and the parent is online on System B, if an instance of the child group faults, the parent is taken offline on System B. After the child has successfully failed over to System C, VCS brings the parent online on System B. If the child cannot fail over, because no suitable system was found, or if the number of online instances of the child group equals the number of systems on which the child group can run, and if one of the child group instances faults, the parent may not be brought online.
online remote soft	Failover parent group soft depends on all instances of the child group being online on another system in the cluster.	Parent can be brought online on any system other than the system on which the child is online. For example if child group is online on SystemA and System C, the parent group can be online on SystemB. When the child faults on SystemA, the parent remains online on SystemB, unless VCS selects SystemB as the target system. After the child successfully fails over to SystemB, VCS brings the parent online on another system, for example, System D.



Dependency Configuration	Definition	Behavior
online remote firm (default)	Failover parent group firm depends on all instances of the child group being online on another system in the cluster.	Failover group can be brought online anywhere as long as an instance of the child group is online on another system. For example, if a child group is online on System A and System C, the parent group can be online on System B. When the child group on System A faults, the parent is taken offline. After the child has successfully failed over to System B, VCS brings the parent online on System D.
offline local	Failover parent group depends on all instances of the child group being offline on the same system, and vice versa.	Failover group can be brought online anywhere as long as an instance of the child group is not online on that system, and vice versa.

Parallel Parent/Failover Child

Dependency Configuration	Definition	Behavior
online global soft	All instances of parent group soft depend on failover group.	An instance of the parent group can be online anywhere as long as the child is online somewhere in the cluster. An instance of the parent group does not fault if an instance of the child group faults.
online global firm (default)	All instances of parent group firm depend on failover group.	An instance of the parent group can be online anywhere as long as the child is online on another system. For example, the child group is online on System A and the parent group is online on System A and System B. When the child faults, all instances of the parent group are taken offline on System A and System B. After the child has successfully failed over to System B, VCS brings all instances of the parent group online on System C and System D.



Dependency Configuration	Definition	Behavior
online remote soft	All instances of parent group soft depend on failover group on any other system.	An instance of the parent group can be online anywhere as long as the child is online on another system. When the child faults, for example, if the child group is online on System A, the parent group can be online on System B and System C, and VCS selects as the target System B on which to bring the child online, the instance of the parent group running on System B is taken offline. After the child has successfully failed over to System B, VCS brings online the failed parent instance to another system, for example, System D.
online remote firm (default)	All instances of parent group firm depend on failover group on any other system.	An instance of the parent group can be online anywhere as long as the child is online on another system. For example, if the child group is online on System A, the parent group can be online on System B and System C. When the child faults, all instances of the parent group are taken offline on System B and System C. After the child has successfully failed over to System B, VCS brings all instances of the parent group online on System A and System D.
offline local	All instances of the parent group depend on the child group being offline on that system and vice versa.	An instance of the parent group can be brought online anywhere as long as the child is not online on the system, and vice versa. For example, if the child group is online on System A, the parent group can be online on System B and System C. If the child faults on System A, and if VCS selects System B as the target on which to bring the child online, the parent on System B is taken offline first.



Parallel Parent/Parallel Child

Dependency Configuration	Definition	Behavior
online local soft	An instance of the parent group soft depends on an instance of the child group on the same system.	<p>An instance of a parent group can be brought online on a system, for example, System A, only if an instance of a child group is online on System A. For example, two instances of the parent are online on System A and System B, and each instance depends on an instance of the child being online on the same system.</p> <p>When the instance of the child group on System A faults, the child group fails over to System C. After the child fails over to another system, the instance of the parent group on System A also fails over to System C. If the child cannot fail over, because no suitable system was found, or because the number of online instances of the child group equals the number of systems on which the child group can run, the parent remains online. Other instances of the parent group are unaffected.</p>
online local firm	An instance of the parent group firm depends on an instance of the child group on the same system.	<p>An instance of a parent group can be brought online on a system, for example, System A, only if an instance of a child group is online on System A. For example, two instances of the parent are online on System A and System B, and each instance depends on an instance of the child being online on the same system.</p> <p>When an instance of the child group on System A faults, the instance of the parent group on System A is taken offline. After the child fails over to another system, for example, System C, VCS brings an instance of the parent group online on System C. Other instances of the parent group are unaffected.</p>



Dependency Configuration	Definition	Behavior
offline local	An instance of a parent group depends on an instance of a child group being offline on the same system and vice versa.	<p>An instance on a system of a parent group can be brought online provided that an instance of the child is not online on the same system and vice versa.</p> <p>For example, if the child group is online on System C and System D, the parent can be online on System A and System G. If the child on System C faults and VCS selects System A as the target on which to bring the child group online, the instance of the parent on System A is taken offline first.</p> <p>When an instance of a child group or parent group faults, it has no effect on the other running instances.</p>



Configuring Service Group Dependencies

To configure a service group dependency, place the `requires` clause in the service group declaration within the VCS configuration file, before the resource dependency specifications, and after the resource declarations. For example:

- ◆ To configure `groupx` and `groupy` as an online local firm dependency:

```
group groupx (...group definition...)...resource declarations...
requires group groupy online local firm...resource dependencies...
```

- ◆ To configure `groupx` and `groupy` as an online global soft dependency:

```
group groupx (...group definition...)...resource declarations...
requires group groupy online global soft...resource dependencies...
```

- ◆ To configure `groupx` and `groupy` as an online remote soft dependency:

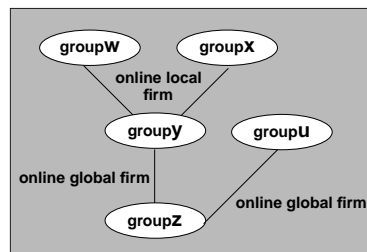
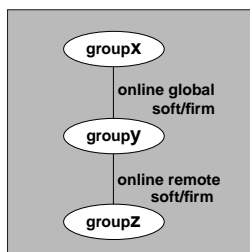
```
group groupx (...group definition...)...resource declarations...
requires group groupy online remote soft...resource dependencies...
```

- ◆ To configure `groupx` and `groupy` as an offline local dependency:

```
group groupx (...group definition...)...resource declarations...
requires group groupy offline local...resource dependencies...
```

Dependency Limitations

Each parent group can link with only one child group; however, a child group can have multiple parents. A service group dependency tree can have three levels, maximum. For example, in the illustration below left, `groupx` requires `groupy` online global, and `groupy` requires `groupz` online remote. In the illustration below right, `groupw` requires `groupy` online local firm, `groupx` requires `groupy` online local firm, `groupy` requires `groupz` online global firm, and `groupu` requires `groupz` online global firm.



Automatic Actions for Service Group Dependencies

Automatic Online

If a service group is configured to start automatically on a system, it is brought online only if the group's dependency requirements are met. This implies that in an online local dependency, parent groups are brought online only after all child groups are brought online.

Automatic Failover

A failover occurs when a service group faults and is migrated to another system. It can also occur when a system crashes and the groups running on that system migrate to other systems in the cluster. For service groups with dependencies, the following actions occur during failover:

- ◆ A target system is selected on which the service group's dependency requirements are met.
- ◆ If a target system exists, but there is a dependency violation between the service group and a parent group, the parent group is migrated to another system to accommodate the service group that is moving to the target system. In conflicts between a child group and a parent group, the child group takes priority.
- ◆ If the service group has a parent with an online local firm dependency, when the child group faults, the parent group is taken offline. When the child successfully fails over to another system, the parent is brought online on that system.
- ◆ If the service group has a parent with an online local soft dependency, when the child group faults, the parent group remains online. When the child successfully fails over to another system, the parent migrates to that system.
- ◆ For soft dependencies, when child group faults and cannot fail over the parent group remains online.
- ◆ For firm dependencies, when child group faults and cannot fail over the parent group remains offline and no further attempt is made to bring it online.



Manual Operations for Service Group Dependencies

As described below, the `hagrp -online`, `-offline`, and `-switch` commands enable you to manually bring a service group online, take it offline, or fail it over.

Manual Online

Basic rules governing how to manually bring a service group online also apply to service groups with dependencies. Additionally, the following rules apply for service groups configured with dependencies. For example:

- ◆ For online dependencies, a parent group cannot be brought online manually if the child is not online.
- ◆ For online local dependencies, a parent group cannot be brought online manually on any system other than the system on which the child is online.
- ◆ For online remote dependencies, a parent group cannot be brought online manually on the system on which the child is online.
- ◆ For offline local dependencies, a parent group cannot be brought online manually on the system on which the child is online.

Typically, bringing a child group online manually is never rejected, except under the following circumstances:

- ◆ For online local dependencies, if parent is online, a child group online is rejected for any system other than the system where parent is online.
- ◆ For online remote dependencies, if parent is online, a child group online is rejected for the system where parent is online.
- ◆ For offline local dependencies, if parent is online, a child group online is rejected for the system where parent is online.

The following examples describe situations where bringing a parallel child group online is accepted:

- ◆ For a parallel child group linked online local with failover/parallel parent, multiple instances of child group online is acceptable.
- ◆ For a parallel child group linked online remote with failover parent, multiple instances of child group online is acceptable, as long as child group does not go online on the system where parent is online.
- ◆ For a parallel child group linked offline local with failover/parallel parent, multiple instances of child group online is acceptable, as long as child group does not go online on the system where parent is online.

Manual Offline

Basic rules governing how to manually take a service group offline also apply to service groups with dependencies. Additionally, VCS rejects manual offlining if the procedure violates existing group dependencies. Typically, firm dependencies are more restrictive to offlining a child group while parent group is online.

- ◆ Parent group offline is never rejected.
- ◆ For all firm dependencies, if parent group is online, child group offline is rejected.
- ◆ For all soft dependencies, child group can be offlined regardless of the state of parent group.

Manual Switch

Switching a service group implies manually taking a service group offline on one system, and manually bringing it back online on another system. Basic rules governing how to manually switch a service group also apply to service group dependencies. Additionally, VCS rejects manual switch if the group does not comply with manual offline or manual online rules described above.



Linking Service Groups (Online/Offline Group Dependencies)

As described previously, a configuration may require that a certain service group be running before another service group can be brought online. For example, a group containing resources of a database service must be running before the database application is brought online.

To specify this dependency, type:

```
hagrps -link parent_group child_group gd_category gd_location gd_type
```

The variable *parent_group* is the name of a service group.

The variable *child_group* is the name of a service group.

The variable *gd_category* is the category of group dependency (online/offline).

The variable *gd_location* is the boundary of *parent_group-child_group* link (local/global/remote).

The optional variable *gd_type* is the type of group dependency (soft/firm).

The *parent_group* is linked to the *child_group* by a link that is described by a combination of *gd_category*, *gd_location* and *gd_type*.

Constraints

- ◆ Each parent group can link with only one child group; however, a child group can have multiple parents.
- ◆ A service group dependency tree can have three levels, maximum.
- ◆ You cannot link two service groups whose current states violate the relationship.
 - ◆ All link requests are accepted if all instances of parent group are offline.
 - ◆ All online local link requests are rejected if for an instance of parent group, an instance of child group is not online on the same system.
 - ◆ All online remote link requests are rejected when an instance of parent group and an instance of child group are running on the same system.
 - ◆ All offline local link requests are rejected when an instance of parent group and an instance of child group are running on the same system.
 - ◆ All link requests are rejected, if parent group is online and child group is offline.
 - ◆ All online global/online remote link requests to link two parallel groups are rejected.
 - ◆ All online local link requests to link a parallel parent group to failover child group are rejected.

VCS Performance Considerations

When you deploy VCS, we recommend that you evaluate performance requirements for:

- ◆ The impact of VCS on overall system performance.
- ◆ The actual VCS performance; for example, the time it takes to failover service groups, etc.

VCS provides various methods that enable you to adjust performance to meet your requirements, including default values that can be used with most applications.

Impact of VCS on Overall System Performance

VCS and its agents run on the same systems as the applications. Therefore, VCS attempts to minimize its impact on overall system performance. The impact of VCS applies to three main components of clustering: the kernel; specifically, GAB and LLT, the VCS engine, and the VCS agents. Each is described below. (For details on attributes or commands mentioned in the following sections, see the chapter on administering VCS from the command line and the chapter on VCS attributes.)

Kernel Components (GAB and LLT)

Typically, overhead of VCS kernel components is minimal. Kernel components primarily provide heartbeat and atomic information exchange among cluster systems. By default, each system in the cluster sends two small heartbeat packets every second to the other systems in the cluster. Heartbeat packets are sent over all network links configured in the `llthosts` configuration file. Intersystem communication also takes place over one of the network links. VCS uses only one network link at a time for intersystem communication, and switches to a different link if the link fails. Typically, these are private network links and do not increase traffic on your public network or LAN. You can configure a public network (LAN) link as low-priority, used only as a heartbeat link, which by default generates one small (approximately 64-byte) broadcast packet per second from each system.



To configure a link as low priority, use the `link-lowpri` command in the `llttab` file. Its parameters are the same as the `link` command. For example:

```
link-lowpri qfe0 /dev/qfe:0 - ether - -
```

VCS Engine

The VCS engine process, `HAD`, runs as a daemon process. By default it runs as a real-time, high-priority process, which ensures that it sends heartbeats to kernel components and responds quickly to any failures. You can adjust the value of VCS engine scheduling class and priority by setting `EngineClass` and `EnginePriority` attributes; however, for optimum performance, we strongly recommend using the default values. VCS also provides a way to control scheduling class and priority for processes invoked by VCS. These can be adjusted by setting `ProcessClass` and `ProcessPriority` attributes. See the chapter on advanced topics for more information on setting these attributes.

VCS “sits” in a loop waiting for messages from agents, `ha` commands, GUI and other systems. Under normal conditions, the number of messages processed by VCS engine is few. They mainly include heartbeat messages from agents and update messages from the global counter. VCS may exchange additional messages when an event occurs, but typically overhead is nominal even during events. (Note that this depends on the type of event; for example, a resource fault may invoke offlining a group on one system and onlining on another system.)

To continuously monitor VCS status, use the VCS GUI, “Cluster Manager,” or the command `hastatus`. Both methods maintain connection to VCS and register for events, and are more efficient compared to running commands like `hastatus -summary` or `hasys` in a loop.

The number of clients connected to VCS can affect performance if several events occur. For example, if five GUI processes are connected to VCS, VCS sends state updates to all five. Maintaining fewer client connections to VCS reduces this overhead.

Agents

Agents have the most impact on overall system performance. VCS agents handle resource type-specific operations. Agents typically provide `online`, `offline`, `monitor`, and `clean` entry points. Entry points can be implemented in C, C++, or a scripting language like shell or Perl. There is one agent per resource type. Agents consist of the agent framework and agent entry points. The agent framework provides common functionality such as communication with VCS engine, multithreading, etc. Follow the performance guidelines below when configuring agents.

Resource Type and Agent Configuration

By default, VCS monitors each resource every 60 seconds. You can change this by modifying the `MonitorInterval` attribute for the resource type. You may consider reducing monitor frequency for non-critical or resources with expensive monitor operations. Note that reducing monitor frequency also means that VCS may take longer to detect a resource fault.

By default, VCS also monitors offline resources. This ensures that if someone brings the resource online outside of VCS control, VCS detects it and flags a concurrency violation for failover groups. To reduce the monitoring frequency of offline resources, modify the `OfflineMonitorInterval` attribute for the type. The VCS agent framework uses multithreading to allow multiple resource operations to run in parallel for the same type of resources. For example, a single `Mount` agent handles all mount resources. Multithreading allows the agent to run the same or different entry points for different resources in parallel. The number of agent threads for each resource type is 10 by default.

To change the default, modify the `NumThreads` attribute for the resource type. The `Mount` agent schedules the `monitor` entry point for all mount resources, based on the `MonitorInterval` or `OfflineMonitorInterval` attributes. If the number of mount resources is more than `NumThreads`, the monitor operation for some mount resources may be required to wait to execute the `monitor` entry point until the thread becomes free.

Additional considerations for the modifying the `NumThreads` attribute include:

- ◆ If you have only one or two resources of a given type, you can set `NumThreads` to a lower value.
- ◆ If you have many resources of a given type, evaluate the time it takes for the `monitor` entry point to execute and the available CPU power for monitoring. For example, if you have 50 mount points, you may want to increase `NumThreads` to get the ideal performance for the `Mount` agent without affecting overall system performance.



Agents typically run with normal scheduling class and priority. You can change this by setting `AgentClass` and `AgentPriority` attributes for given resource type. This can be used to designate high-priority monitoring for critical resources, and low-priority monitoring for non-critical resource types.

If agent entry points are written in scripting language, their scheduling class and priority are normal by default. They can be changed by modifying the `ScriptClass` and `ScriptPriority` attributes for the given resource type. When you develop agents, consider the following:

- ◆ If you write a custom agent, write the monitor entry point using C or C++. If you write a script-based monitor, VCS must invoke a new process each time with the monitor. This can be costly if you have many resources of that type.
- ◆ If monitoring the resources type is proving costly, you can divide it into “shallow” and “deep” monitoring. Shallow monitoring checks only for the existence of the process. Deep monitoring performs detailed checking of the resource; for example, deep monitoring a database service can send an SQL query. Whether to use shallow or deep monitoring depends on your configuration requirements. You may decide to perform one deep monitoring after every three shallow monitorings, or decide to make it configurable.

Note Agents attempt to minimize communication to VCS. Agents schedule periodic monitor operations, and only notify the VCS engine of changes to a resource state.

Additional Considerations for Agents

Properly configure the attribute `SystemList` for your service group. For example, if you know that a service group can go online on `sysa` and `sysb` only, *do not* include other systems in the `SystemList`. This saves additional agent processes and monitoring overhead.

VCS Performance

This section describes factors that affect VCS operations, such as bringing a resource or service group online, taking them offline, and failing over service groups over to a different system.

Cluster Boot Time

When a cluster system boots, the kernel drivers and VCS process are started in a particular order. If it is the first system in the cluster, VCS reads the cluster configuration file `main.cf` and builds an “in-memory” configuration database. This is the `LOCAL_BUILD` state. When the system finishes building the configuration database, it transitions into the `RUNNING` mode. If another system joins the cluster while the first system is in the `LOCAL_BUILD` state, it must wait until the first system transitions into `RUNNING` mode. The time it takes to build the configuration depends on the size of the configuration and the dependencies. VCS creates an object for each system, service group, type, and resource. Typically, the number of systems, service groups and types are few, so the number of resources and resource dependencies determine how long it takes to build the configuration database and to get VCS into `RUNNING` mode. If a system joins a cluster in which at least one system is in `RUNNING` mode, it builds the configuration from the lowest system in that mode.

Note Onlining service groups as part of AutoStart occurs after VCS transitions to `RUNNING` mode.

Bringing a Resource Online

The `online` entry point of an agent attempts to bring the resource online. This entry point may return before the resource is fully online. The subsequent monitor determines if the resource is online, then reports that information to VCS. The time it takes to bring a resource online equals the time for the resource to go online, plus the time for the subsequent monitor to execute and report to VCS.

Most resources are online when the `online` entry point finishes. The agent schedules the monitor immediately after the entry point finishes, so the first monitor detects the resource as online. For some resources, such as a database server, recovery can take longer, so the time it takes to bring a resource online depends on the amount of data to recover.



Taking a Resource Offline

Similar to the `online` entry point, the `offline` entry point attempts to offline the resource, and may return before resource is actually offline. Subsequent monitoring determines whether the resource is offline or not. The time it takes to offline a resource equals the duration of the subsequent monitor executions plus its reporting to VCS that the resource is offline. Most resources are typically offline when the offline entry point finishes. The agent schedules the monitor immediately after the `offline` entry point finishes, so the first monitor detects the resource as offline.

Bringing a Service Group Online

The time it takes to bring a service group online depends on the number of resources in the service group, the service group topology, and the time to online the group's resources. For example, if a service group G1 has three resources, R1, R2, and R3 (where R1 depends on R2 and R2 depends on R3), VCS first onlines R3. When R3 is online, VCS onlines R2. When R2 is online, VCS onlines R1. The time it takes to online G1 equals the time it takes to bring all resources online. However, if R1 depends on both R2 and R3, but there was no dependency between them, the online operation of R2 and R3 is started in parallel. When both are online, R1 is brought online. The time it takes to online the group is Max (the time to online R2 and R3), plus the time to online R1. Typically, broader service group trees allow more parallel operations and can be brought online faster. Deeper service group trees do not allow much parallelism and serializes the group online operation.

Taking a Service Group Offline

The time it takes to offline a service group depends on the number of resources in the service group, the service group topology, and the time to offline the group's resources. Service group offlining works from the top down, as opposed to onlining, which works from the bottom up.

Detecting Resource Failure

The time it takes to detect a resource fault or failure depends on the `MonitorInterval` attribute for the resource type. When a resource faults, the next monitor detects it. The agent may not declare the resource as faulted if the `ToleranceLimit` attribute is set to non-zero. If the `monitor` entry point reports offline more often than the number set in `ToleranceLimit`, the resource is declared faulted. However, if the resource remains online for the interval designated in `ConfInterval`, any earlier reports of offline are not counted against `ToleranceLimit`.

When the agent determines that the resource is faulted, it calls the `clean` entry point, if implemented. This is done to verify that the resource is completely offline. The next monitor after `clean` confirms the offline. The agent then tries to online the resource again if `RestartLimit` is non-zero. The agent attempts to restart the resource according to the number set in `RestartLimit` before it gives up and informs the VCS engine that the resource is faulted. However, if the resource remains online for the interval designated in `ConfInterval`, earlier attempts to restart are not counted against `RestartLimit`.

In most cases, `ToleranceLimit` is 0. The time it takes to detect a resource failure is the time it takes the agent monitor to detect failure, plus the time to clean up the resource if the `clean` entry point is implemented. Therefore, the time it takes to detect failure depends on the `MonitorInterval`, the efficiency of the `monitor` and `clean` (if implemented) entry points, and the `ToleranceLimit` (if set).

In some cases, the failed resource may hang and may also cause the monitor to hang. For example, if the database server is hung and the monitor tries to query, the monitor will also hang. If the `monitor` entry point is hung, the agent eventually times it out. By default, the agent timeouts the `monitor` entry point after 60 seconds. This can be adjusted by changing the `MonitorTimeout` attribute. The agent retries `monitor` after the `MonitorInterval`. If the `monitor` entry point times out consecutively for the number of times designated in the attribute `FaultOnMonitorTimeouts`, the agent treats the resource as faulted. The agent calls `clean`, if implemented. The default value of `FaultOnMonitorTimeouts` is 4, and can be changed according to the type. A high value of this parameter delays detection of a fault if the resource is hung. If the resource is hung and causes the `monitor` entry point to hang, the time to detect it depends on `MonitorTimeout`, `FaultOnMonitorTimeouts`, `monitor` and `clean` (if implemented) efficiency.

Detecting System Failure

When a system crashes or is powered off, it stops sending heartbeats to other systems in the cluster. By default, other systems in the cluster wait 21 seconds before declaring it dead. The time of 21 seconds derives from 16 seconds default timeout value for LLT peer inactive timeout, plus 5 seconds default value for GAB stable timeout. LLT peer inactive timeout value can be changed by setting LLT peer-inact value in `llthosts` file. The default peer inactive timeout is 16 seconds, and can be modified in the `llttab` file. For example, to specify 12 seconds, type:

```
set-timer peerinact:1200
```

GAB stable timeout can be changed by specifying:

```
gabconfig -t timeout_value_milliseconds
```

Though this can be done, we *do not* recommend changing the values of the LLT peer inactive timeout and GAB stable timeout.



Note If a system reboots, it becomes unavailable until the reboot is complete. The reboot process kills all processes including the VCS engine process. When the VCS process is killed, other systems in the cluster mark all service groups that can go online on the rebooted system as autodisabled. The AutoDisabled flag is cleared when the system goes offline. As long as the system goes offline within 60 seconds, VCS treats this as a system reboot. This default value can be changed by modifying the ShutdownTimeout attribute.

Detecting Network Link Failure

If a system loses a network link to the cluster, other systems stop receiving heartbeats over the links from that system. As mentioned above, LLT detects this, and waits for 16 seconds before declaring that the system lost a link.

Service Group Switch

The time it takes to switch a service group equals the time to offline a service group on the source system, plus the time to bring the service group online on the target system.

Service Group Failover

The time it takes to failover a service group when a resource faults equals the time to detect the resource fault, plus the time to offline the service group on source system, plus the time for the VCS policy module to select target system, plus the time to bring the service group online on target system.

The time it takes to failover a service group when a system faults equals the time to detect system fault, plus the time to offline the service group on source system, plus the time for the VCS policy module to select target system, plus the time to bring the service group online on target system.

The time it takes the VCS policy module to detect the target system is negligible in comparison to the other factors. If you have a firm group dependency and the child group faults, VCS offlines all immediate and non-immediate parent groups before bringing the child group online on the target system.

Event Notification

VCS provides a method for notifying the administrator of important events such as a resource or system fault. This method is known as event notification, or *event triggers*. This feature also enables the administrator to take specific actions in response to particular events. Event triggers are defined on page 158.

How VCS Performs Event Notification

- ✓ VCS determines if the event is enabled.
- ✓ VCS invokes `hatrigger`.

VCS calls `hatrigger`, a high-level Perl script located at:

- ◆ On **UNIX**: `$VCS_HOME/bin/hatrigger`
- ◆ On **Windows NT**: `VCS_HOME\bin\hatrigger.pl`

VCS also passes the name of event trigger and the parameters specific to the event. For example, when a service group becomes fully online on a system, VCS invokes `hatrigger -postonline system service_group`. Note that VCS does not wait for `hatrigger` or the event trigger to complete execution. After calling the triggers, VCS continues normal operations.

Event triggers are invoked on the system where the event occurred, with the following exceptions:

- ◆ The InJeopardy, SysOffline, and NoFailover event triggers are invoked from the lowest-numbered system in `RUNNING` state.
 - ◆ The Violation event trigger is invoked from all systems on which the service group was brought partially or fully online.
- ✓ The script `hatrigger` invokes an event trigger.

The script `hatrigger` performs actions common to all triggers, and calls the intended event trigger as instructed by VCS. This script also passes the parameters specific to the event.



Event triggers are invoked according to following conventions:

- ◆ On **UNIX**: by event names; for example, `violation` to denote a concurrency violation.
- ◆ On **Windows NT**: by event names, followed by the extension `.exe`, `.pl`, `.ksh`, or `.bat`; for example, `violation.pl`.

Sample Scripts

VCS provides sample Perl script for each event trigger. These scripts can be customized according to your requirements. On UNIX, you may also write your own Perl script, C, or C++ programs instead of using the sample scripts. On Windows NT, you may write your own Perl script only.

Sample Perl scripts for event triggers are located in the following directories:

- ◆ On **UNIX**: `$VCS_HOME/bin/sample_triggers`
- ◆ On **Windows NT**: `VCS_HOME\bin\sample_triggers`

Types of Event Triggers

The following table describes the various event triggers, including their usage, parameters, and location.

Note that event triggers must reside on all systems in the cluster in the following directories:

- ◆ On **UNIX**: `$VCS_HOME/bin/triggers`
- ◆ On **Windows NT**: `VCS_HOME\bin\triggers`

Note If VCS determines that there is no corresponding trigger script or executable in the locations listed for each event trigger, VCS takes no further action.

InJeopardy Event Trigger

Usage	Location
<p><code>- injeopardy system system_state</code></p> <p>The variable <i>system</i> represents the name of the system.</p> <p>The variable <i>system_state</i> represents the value of the State attribute. See “System States” on page 193.</p>	<ul style="list-style-type: none"> ◆ UNIX: /opt/VRTSvcs/bin/triggers/injeopardy ◆ Windows NT: VCS_HOME\bin\triggers\InJeopardy.<i>extension</i> (exe., .pl, .ksh, or .bat)
<p>Description</p>	
<p>Invoked when a system is in jeopardy. Specifically, this trigger is invoked when a system has only one remaining link to the cluster, and that link is a network link (LLT). This is a considered a critical event because if the system loses the remaining network link, VCS does not fail over the service groups that were online on the system. Using this trigger to notify the administrator of the critical event enables the administrator to take appropriate action to ensure that the system has at least two links to the cluster.</p> <p>This event trigger is non-configurable.</p>	

NoFailover Event Trigger

Usage	Location
<p><code>- nofailover system service_group</code></p> <p>The variable <i>system</i> represents the name of the last system on which an attempt was made to online the service group.</p> <p>The variable <i>service_group</i> represents the name of the service group.</p>	<ul style="list-style-type: none"> ◆ UNIX: /opt/VRTSvcs/bin/triggers/nofailover ◆ Windows NT: VCS_HOME\bin\triggers\NoFailover.<i>extension</i> (exe., .pl, .ksh, or .bat)
<p>Description</p>	
<p>Called from the lowest-numbered system in RUNNING state when a service group cannot fail over.</p> <p>This event trigger is non-configurable.</p>	



PostOffline Event Trigger

Usage	Location
<p><code>- postoffline system service_group</code></p> <p>The variable <i>system</i> represents the name of the system.</p> <p>The variable <i>service_group</i> represents the name of the service group that went offline.</p> <p>Default = 0</p>	<ul style="list-style-type: none"> ◆ UNIX: /opt/VRTSvcs/bin/triggers/postoffline ◆ Windows NT: VCS_HOME\bin\triggers\PostOffline.<i>extension</i> (exe., .pl, .ksh, or .bat)
<p>Description</p>	
<p>This event trigger is invoked on the system where the group went offline from a partial or fully online state. This trigger is invoked when the group faults, or is taken offline manually.</p> <p>This event trigger is non-configurable.</p>	

PostOnline Event Trigger

Usage	Location
<p><code>- postonline system service_group</code></p> <p>The variable <i>system</i> represents the name of the system.</p> <p>The variable <i>service_group</i> represents the name of the service group that went online.</p> <p>Default = 0</p>	<ul style="list-style-type: none"> ◆ UNIX: /opt/VRTSvcs/bin/triggers/postonline ◆ Windows NT: VCS_HOME\bin\triggers\PostOnline.<i>extension</i> (exe., .pl, .ksh, or .bat)
<p>Description</p>	
<p>This event trigger is invoked on the system where the group went online from a partial or fully offline state. This event trigger is non-configurable.</p>	

PreOnline Event Trigger

Usage	Location
<pre data-bbox="149 309 521 364">- preonline system service_group whyonlining</pre> <p data-bbox="149 390 578 447">The variable <i>system</i> represents the name of the system.</p> <p data-bbox="149 460 578 572">The variable <i>service_group</i> represents the name of the service group on which the <code>hagrps</code> command was issued or the fault occurred.</p> <p data-bbox="149 586 564 642">The variable <i>whyonlining</i> represents two values:</p> <p data-bbox="149 656 549 737">FAULT indicates that the group was brought online in response to a group failover or switch.</p> <p data-bbox="149 751 564 833">MANUAL indicates that group was brought online manually on the system represented by the variable <i>system</i>.</p>	<ul style="list-style-type: none"> <li data-bbox="605 303 1135 329">◆ UNIX: /opt/VRTSvcs/bin/triggers/preonline <li data-bbox="605 355 1299 416">◆ Windows NT: VCS_HOME\bin\triggers\PreOnline.<i>extension</i> (exe., .pl, .ksh, or .bat)
<h3>Description</h3>	
<p data-bbox="149 920 1313 1003">Indicates that the VCS engine should not online a service group in response to an <code>hagrps -online</code> command or a fault. The engine should instead call a user-defined script that checks for external conditions before bringing the group online.</p> <p data-bbox="149 1015 1313 1072">Note If it is OK to bring the group online, it is then the responsibility of the PreOnline event trigger to bring the group online using the format: <code>hagrps -online -nopre service_group -sys system</code></p> <p data-bbox="149 1085 878 1111">If the trigger does not exist, VCS continues to bring the group online.</p> <p data-bbox="149 1123 956 1149">If you do want to bring the group online, define the trigger to take no action.</p> <p data-bbox="149 1161 506 1187">This event trigger is configurable.</p> <ul style="list-style-type: none"> <li data-bbox="149 1199 1035 1258">◆ To enable this trigger, specify <code>PreOnline=1</code> within the group definition, or use: <code>hagrps -modify service_group PreOnline 1</code> <li data-bbox="149 1270 1035 1329">◆ To disable the trigger, specify <code>PreOnline=0</code> within the group definition, or use: <code>hagrps -modify service_group PreOnline 0</code> 	



ResFault Event Trigger

Usage	Location
<p data-bbox="104 309 422 335"><code>- resfault <i>system resource</i></code></p> <p data-bbox="104 361 529 418">The variable <i>system</i> represents the name of the system.</p> <p data-bbox="104 430 479 487">The variable <i>resource</i> represents the name of the faulted resource.</p>	<ul data-bbox="556 303 1236 418" style="list-style-type: none"><li data-bbox="556 303 1068 331">◆ UNIX: /opt/VRTSvcs/bin/triggers/resfault<li data-bbox="556 357 1236 418">◆ Windows NT: VCS_HOME\bin\triggers\ResFault.<i>extension</i> (exe., .pl, .ksh, or .bat)
Description	
<p data-bbox="104 569 1243 656">Invoked on the system where an online resource has faulted. (The resource has transitioned from ONLINE to FAULTED.) Note that when a resource is faulted, resources within the upward path of the faulted resource are also brought down.</p> <p data-bbox="104 668 508 696">This event trigger is non-configurable.</p>	

ResNotOff Event Trigger

Usage	Location
<p>- <code>resnotoff system resource</code></p> <p>The variable <code>system</code> represents the name of the system.</p> <p>The variable <code>resource</code> represents the name of the resource that failed to go offline.</p>	<ul style="list-style-type: none"> ◆ UNIX: <code>/opt/VRTSvcs/bin/triggers/resnotoff</code> ◆ Windows NT: <code>VCS_HOME\bin\triggers\ResNotOff.extension</code> (exe., .pl, .ksh, or .bat)
<h3>Description</h3>	
<p>This trigger is invoked when resource is unable to go offline under the following circumstances:</p> <ul style="list-style-type: none"> ◆ manual group or resource offline ◆ manual group switch ◆ group failover ◆ This trigger is also invoked when a resource is online and the monitor hangs consecutively the number of times designated in the attribute <code>FaultOnMonitorTimeouts</code>, then hangs again after VCS calls the <code>clean</code> entry point. <p>ResNotOff provides a way to program events such as a resource unable to go offline. If the resource does not go offline when a service group faults, VCS cannot perform failover. If this happens with a critical service group, consider rebooting the machine on which the resource is unable to go offline.</p> <p>VCS provides a way to attach a priority to the group. VCS itself does not interpret the priority, but the trigger can make use of it. This is an attribute set by the user, which designates the priority associated with the service group. For example, you can select the value 1 to denote the highest priority. (See “Priority” on page 209.)</p> <p>To program ResNotOff to handle the case of a resource unable to go offline for a critical service group:</p> <ol style="list-style-type: none"> 1. From resource name, identify the service group and its priority, if applicable. 2. If other critical service groups are online on the node where the resource is unable to go offline, switch them to another node using the command <code>hagr -switch</code>. 3. Reboot the machine on which the resource is unable to go offline. <p>To handle a case of a resource unable to go offline for a non-critical service group, use this trigger to send an email to the administrator notifying him or her of the event.</p> <p>This event trigger is non-configurable.</p>	



SysOffline Event Trigger

Usage	Location
<pre>- sysoffline system system_state</pre> <p>The variable <i>system</i> represents the name of the system.</p> <p>The variable <i>system_state</i> represents the value of the State attribute. See “System States” on page 193.</p>	<ul style="list-style-type: none"> ◆ UNIX: /opt/VRTSvcs/bin/triggers/sysoffline ◆ Windows NT: VCS_HOME\bin\triggers\SysOffline.<i>extension</i> (exe., .pl, .ksh, or .bat)
Description	
<p>Called from the lowest-numbered system in RUNNING state when a system leaves the cluster.</p> <p>This event trigger is non-configurable.</p>	

Violation Event Trigger

Usage	Location
<pre>- violation system service_group</pre> <p>The variable <i>system</i> represents the name of the system.</p> <p>The variable <i>service_group</i> represents the name of the service group that was fully or partially online.</p>	<ul style="list-style-type: none"> ◆ UNIX: /opt/VRTSvcs/bin/triggers/violation ◆ Windows NT: VCS_HOME\bin\triggers\Violation.<i>extension</i> (exe., .pl, .ksh, or .bat)
Description	
<p>This trigger is invoked only on the system that caused the concurrency violation. Specifically, it takes the service group offline on the system where the trigger was invoked. Note that this trigger applies to failover groups only. The default trigger takes the service group offline on the system that caused the concurrency violation. See page 11 for more information.</p> <p>This event trigger is non-configurable.</p>	

Scheduling Class and Priority Configuration Support

VCS allows you to specify priorities and scheduling classes for VCS processes. VCS supports the following scheduling classes:

- ◆ RealTime (specified as “RT” in the configuration file)
- ◆ TimeSharing (specified as “TS” in the configuration file)

Additional Information for Windows NT Users

On Windows NT, RT is mapped to HIGH_PRIORITY_CLASS and TS is mapped to NORMAL_PRIORITY_CLASS.

Priority Ranges

The following table displays platform-specific priority range for RealTime and TimeSharing processes.

Platform	Scheduling Class	Default Priority Range Weak / Strong	Priority Range Using #ps Commands
Solaris	RT	0 / 59	100 / 159
	TS	-60 / 60	N/A Note On Solaris, use #ps -ae 0 pri, args
HP-UX	RT	127 / 0	127 / 0
	TS	N/A	N/A Note On HP-UX, use #ps -ae1



Default Scheduling Classes and Priorities

The following table lists the default class and priority values used by VCS. Note that the default priority value is platform-specific. Therefore, when priority is set to "" (empty string), VCS converts the priority to a value specific to the platform on which the system is running. For TS, the default priority equals the strongest priority supported by the TimeSharing class. For RT, the default priority equals two less than the strongest priority supported by the RealTime class. So, if the strongest priority supported by the RealTime class is 59, the default priority for the RT class is 57.

Process	Default Scheduling Class	Default Priority	
		Solaris	HP-UX
Engine	RT	57 (Strongest - 2)	2 (Strongest + 2)
Process created by Engine	TS	60 (Strongest)	N/A
Agent	TS	60 (Strongest)	N/A
Script	TS	60 (Strongest)	N/A

For information on cluster and resource attributes included with the scheduling feature, and how to set them, refer to [Appendix C](#).

VCS Security

Issuing Commands from the Command-Line Interface and Cluster Shell

Command issued from the VCS command-line interface, hacli, or from the Cluster Shell in the VCS graphical user interface carry the full privilege of “root” and “administrator” of the host environment. In UNIX, commands executed on cluster member hosts have `uid=0`. To disable this function, substitute `hacli_runcmd` in the VCS installation directory with a blank script on each of the cluster member hosts.

Controlling Connections to VCS

The system attribute `GUIPAddr` can be used to control access to VCS. This attribute determines the local IP address that VCS uses to accept connections. Incoming connections over any other IP addresses are dropped.

If the requirement is to reject any external unsecure connections over local IP addresses, the attribute `GUIIPAddr` can be set to `localhost` (127.0.0.1). With this setting, VCS will not accept any external connections. Secure sockets can then be used on the client and server to send or receive encrypted data. The `ssh` layer on the client can be used to encrypt and forward data to the `ssh` layer on the server, which then forwards the data to the VCS engine (on `localhost`).

Handling Network Failure

VCS protects against network partitions by requiring that all systems be connected by two or more communication channels. In a VCS cluster, all systems send heartbeats to each other across communication channels. If a system's heartbeats are not received across one channel, VCS detects that the channel has failed. If a system's heartbeats are not received across any channels, VCS detects that the system has failed. The services running on that system are then restarted on another.

VCS continues to operate as a single cluster when at least one network channel exists between the systems. However, when only one channel remains, failover due to system failure is disabled. Even after the last network connection is lost, VCS continues to operate as partitioned clusters on each side of the failure.

Note For more information on protecting your cluster against network failure, see the section in the *VERITAS Cluster Server Installation Guide* regarding GAB verification.

About Cluster Memberships

In VCS, memberships are sets of systems participating in the cluster. There are different types of memberships. A regular membership constitutes systems that communicate with each other across one or more network channels.

A *jeopardy membership* is distinct from the regular membership of the cluster and has two components:

- ◆ Systems in the regular membership, that are not part of the largest subset of systems, connected to each other by at least two channels.
- ◆ Systems outside the regular membership that are still active and writing heartbeats to disk.

Note Disks alone cannot provide the levels of communication service required for systems to participate as a single VCS cluster.

Updates to the jeopardy membership are written to the system logs:

- ◆ On **UNIX**: `/var/adm/messages` or `/var/adm/syslog/syslog.log`



- ◆ On **Windows NT**: See the Event Viewer.

Updates are available from the output of the command `gabconfig -a`.

Disabling Failover

When VCS loses communication with a system, a new regular membership is issued that excludes the departed system. VCS must then determine if it should restart that system's services, or if the system is running services outside of communication with VCS. Two conditions indicate that the system could still be running the services:

- ◆ Prior to the system's departure, the systems remaining in the new membership were connected to the departed system by only one communication channel.
- ◆ The departed system continues writing heartbeats to disk.

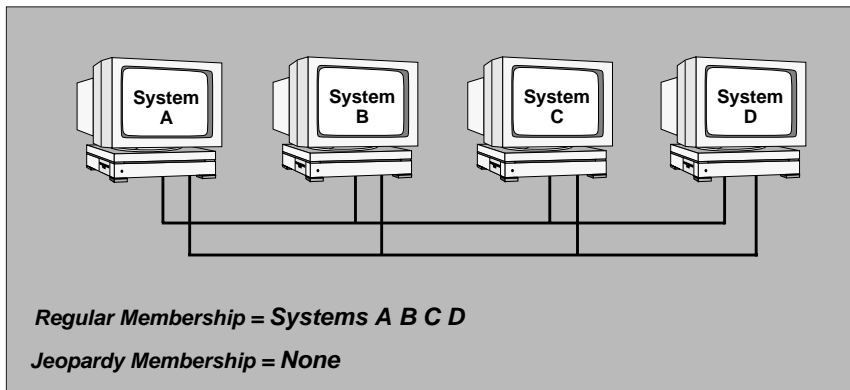
VCS detects these conditions using the jeopardy membership. If there is at least one system in the new regular membership that was not part of the prior jeopardy membership, then failover is disabled only for those systems that left the regular membership and were part of the prior jeopardy membership. Failover is also disabled for systems that are in the new jeopardy membership and outside of the new regular membership. This indicates these systems are actively writing heartbeats to disk.

If there are no systems in the new regular membership that were not part of the previous jeopardy membership, failover is disabled for all systems that have departed. This indicates that connections from the remaining systems to all systems in the prior regular membership were potentially unreliable.

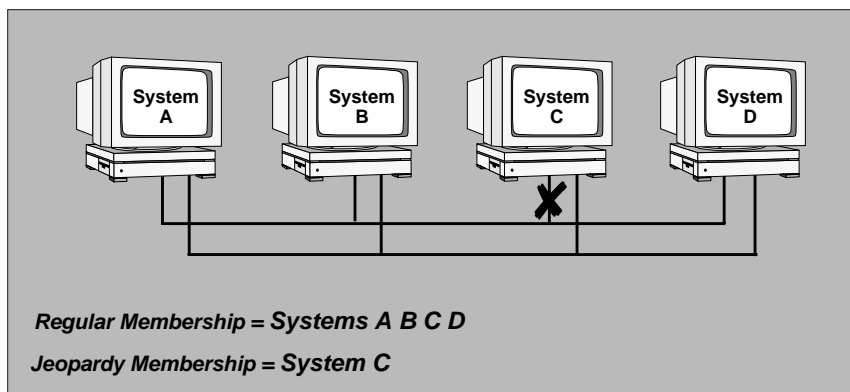


Example of How VCS Handles Network Failure

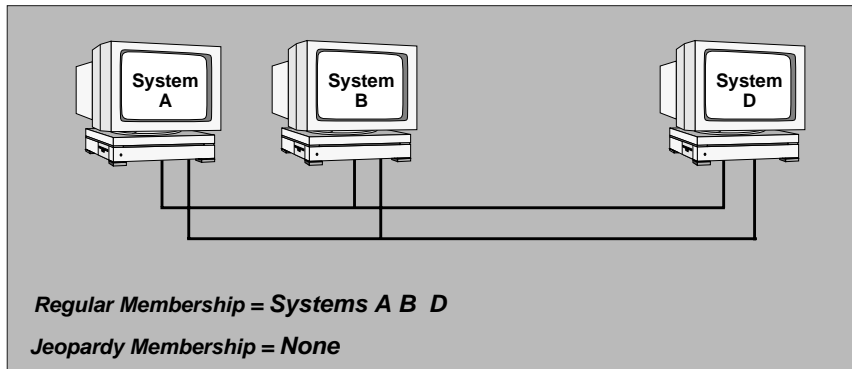
In the following example, a single cluster has two networks connecting four systems: A, B, C, and D.



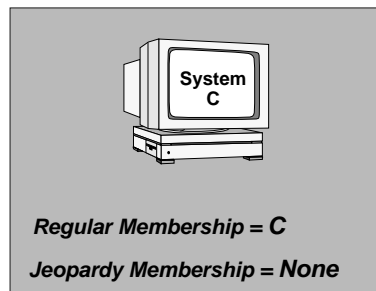
If a network card on C fails, a jeopardy membership for C is issued, along with a regular membership for systems A, B, C, and D.



If the remaining network card on C fails, VCS forms two clusters: one with a membership of systems A, B, and D (illustrated below), and the other with a membership of C only. The cluster of systems A, B, and D disable failover for services running on C because it was part of the prior jeopardy membership.



The cluster consisting only of C disables failover for the services running on systems A, B, and D because they have left the membership, and because the new regular membership does not include systems that were not in the prior jeopardy membership.



▼ To restore network connections

1. Shutdown VCS on System C. (See “[Stopping VCS](#)” on page 49.)
2. Reconnect the private networks.
3. Wait for GAB seed port membership (port a), including all systems to appear on the system console (or `gabconfig -a`).
4. Restart VCS on System C. (See “[Starting VCS](#)” on page 47.)









Starting VCS on Windows NT

If VCS fails to start automatically during the installation procedure, perform the following steps on each system in the cluster:

▼ From the Windows NT Services Applet

1. Double-click the Services icon in the Control Panel.
2. Select VERITAS High Availability Engine.
3. Click Start.
4. Click Close.

▼ From the Command Line

```
C:\> net start had
```

or

```
C:\> hastart
```

To start Command Server, type:

```
C:\> net start cmdserver
```

Note To specify command-line arguments to HAD, you must use the Windows NT Services applet. Enter the arguments in the Startup Parameters text field.



Startup Errors

When VCS is started on a system, it checks the state of its local configuration file and registers with GAB for cluster membership. If the local configuration is valid, and if no other system is running VCS, it builds its state from the local configuration file and enters the RUNNING state. The following steps are then displayed on the console:

```
VCS: starting on: system1
VCS: waiting for configuration status
VCS: local configuration valid
VCS: registering for cluster membership
VCS: waiting for cluster membership
GAB: Port h gen da9f0001 membership ;1
VCS: received new cluster membership
VCS: building from local configuration
VCS: entering RUNNING state
```

Occasionally, you may encounter problems when trying to start VCS. The following information describes some of these problems and provides solutions.

Local Configuration Is Invalid

The following messages indicate that the local VCS configuration is invalid:

```
VCS: local configuration missing
VCS: local configuration invalid
VCS: local configuration stale
```

To correct this, start the VCS engine on another system that has a valid configuration file. The system with the configuration error “pulls” the valid configuration from the other system.

Another method is to correct the configuration file on the local system and force VCS to reread the configuration file. If the file appears valid, verify that is not an earlier version. It is possible that VCS marked the configuration stale by creating a `.stale` file because the last VCS shutdown was not graceful.

The `.stale` files are created in the following locations:

- ◆ On **UNIX**: `/etc/VRTSvcs/conf/config/.stale`
- ◆ On **Windows NT**: `%VCS_HOME%\conf\config`

Type the following commands to verify the configuration and force VCS to reread the configuration file:

- ◆ On **UNIX**:

```
# cd /etc/VRTSvcs/conf/config
# hacf -verify
# hasys -force system
```

- ◆ On **Windows NT**:

```
C:\> cd \Program Files\VERITAS\conf\config
C:\> hacf -verify
C:\> hasys -force system
```

VCS Can't Register for Cluster Membership

The following message indicates that GAB was not registered or has become unregistered:

```
VCS: registration failed. Exiting
```

GAB is registered by the `gabconfig` command in the following files:

- ◆ On **UNIX**: `/etc/gabtab`
- ◆ On **Windows NT**: `%VCS_ROOT%\comms\gab\gabtab.txt`

Verify that the file exists and that it contains the command `gabconfig -c`. Re-execute the scripts `/etc/rc2.d/S92gab` or `/sbin/rc2.d/S92gab`.

GAB can become unregistered if LLT is set up incorrectly. Verify that the file is correct:

- ◆ On **UNIX**: `/etc/llttab`
- ◆ On **Windows NT**: `%VCS_ROOT%\comms\llt\llttab.txt`

If the LLT configuration is incorrect, make the appropriate changes and reboot. See the accompanying installation guide for more information on configuring LLT.



VCS Hangs at Message “Waiting for cluster membership.”

This indicates that GAB may not be seeded. If this is the case, the command `gabconfig -a` does not show any members, and the following messages may appear on the console or in `/var/adm/messages` on UNIX, and in the Event Viewer on Windows NT:

```
GAB: Port a registration waiting for seed port membership
GAB: Port h registration waiting for seed port membership
```

There are two reasons for receiving these messages:

- ◆ GAB is waiting for the number of systems specified by the `-n` option of `gabconfig` to start. Typically, this command is in `/etc/gabtab` on UNIX, and `%VCS_ROOT%\comms\gab\gabtab.txt` on Windows NT.
- ◆ If the `-n` option is not specified, GAB is waiting for a system to start that has the `-x` option specified to `gabconfig`.

You can issue `gabconfig -cx` on any system waiting for seed-port membership. See page 184 for more information on seeding the cluster.

Network Partitioning

With VCS, two or more communication channels guard against network partitioning; a condition where a failure on the network is misinterpreted as a failure of one or more systems in the cluster. If one system in the cluster assumes wrongly that another system has failed, it may restart applications already running on the other system, thereby corrupting the data.

Using a second communication channel enables VCS to distinguish between network and system failures. If all but one network channel fails, VCS enters a degraded mode that disables automatic application failover caused by system failure. If the last network channel fails, VCS partitions into multiple “mini-clusters” without failing over or shutting down applications. This design enables administrative services to operate uninterrupted; for example, you can use VCS to shut down applications during system maintenance.

Reconnecting the Private Network

If the private network has been disconnected, you must shutdown VCS *before* reconnecting the systems. (See “[Stopping VCS](#)” on page 49). Failure to do so results in one or more systems being halted until only the larger of the previously disconnected mini-clusters remains. Halting the systems protects the integrity of shared storage when network connections become unstable. In such an environment, the data on shared storage may already be corrupted by the time the network connections are stabilized.

To forego this protection, add the option `-j` to the `gabconfig` command. This option disables system halt when rejoining a cluster after a split.

The `gabconfig` command is located:

- ◆ On UNIX, in `/etc/gabtab`
- ◆ On Windows NT, in `%VCS_ROOT%\comms\gab\gabtab.txt`

Network Partitions and the UNIX Boot Monitor

Most UNIX systems provide a console-abort sequence that enables you to halt and continue the processor. Continuing operations after the processor has stopped may corrupt data and is therefore unsupported by VCS. Specifically, when a system is halted with the abort sequence it stops producing heartbeats. The other systems in the cluster then consider the system failed and take over its services. If the system is later enabled with another console sequence, it continues writing to shared storage as before, even though its applications have been restarted on other systems where available.

We recommend disabling the console-abort sequence. See the accompanying installation guide for instructions.

Preexisting Network Partitions

A *preexisting network partition* refers to failures in communication channels that occur while the systems are down. Regardless of whether the cause is scheduled maintenance or system failure, VCS cannot respond to failures when systems are down. This leaves VCS vulnerable to network partitioning when the systems are booted.



VCS Seeding

To protect your cluster from a preexisting network partition, VCS employs the concept of a *seed*. By default, when a system comes up, it is not *seeded*. (See the accompanying installation guide for instructions on changing the default behavior.) Systems can be seeded automatically or manually. Note that only systems that have been seeded can run VCS.

Systems are seeded automatically in one of two ways:

- ◆ When an unseeded system communicates with a seeded system.
- ◆ When all systems in the cluster are unseeded and able to communicate with each other.

VCS requires that you declare the number of systems that will participate in the cluster. When the last system is booted, the cluster will seed and start VCS on all systems. Systems can then be brought down and restarted in any combination. Seeding is automatic as long as at least one instance of VCS is running somewhere in the cluster. See the accompanying installation guide for instructions on seeding your cluster.

Bringing Service Groups Online

Occasionally, you may encounter problems when trying to bring VCS service groups online. The following information describes some of these problems and provides solutions.

Incorrect Local Name for System - UNIX Only

A service group cannot be brought online if VCS has an incorrect local name for the system. This occurs when the name returned by the command `uname -n` does not match the system name in the file `main.cf`. This is typically the case when `uname -n` returns a fully domain-qualified name. When this happens, type `hasys -list` on the system to display the fully domain-qualified name and the name in `main.cf`.

To correct this problem, create the following file:

```
/etc/VRTSvcs/conf/sysname
```

Place the file in the system name as it appears in `main.cf`. Shut down VCS using the command `hastop -local`, and restart VCS.

System Not in RUNNING State

Type `hasys -display system` to verify the value of the `SysState` attribute and to verify that the system is running. See page 193 for more information on system states.

Service Group Not Configured to Run on the System

The `SystemList` attribute of the group may not contain the name of the system. Use the output of the command `hagrp -display service_group` to verify the system name.

Service Group Not Configured to AutoStart

If the service group is not starting automatically on the system, the group may not be configured to `AutoStart`, or may not be configured to `AutoStart` on that particular system. Use the output of the command `hagrp -display service_group` to verify the values of the `AutoStart` and `AutoStartList` attributes.

Service Group AutoDisabled

When VCS does not know the status of a service group on a particular system, it autodisables the service group on that system. Autodisabling occurs under the following conditions:

- ◆ When the VCS engine is not running on the system.
- ◆ When all resources within the service group are not probed on the system.
- ◆ When a particular system is visible through disk heartbeat only.

Under these conditions, all service groups that include the system in their `SystemList` attribute are autodisabled. *Note that this does not apply to systems that are powered off.*

Use the output of the command `hagrp -display service_group` to verify the value of the `AutoDisabled` attribute.

Caution To bring a group online manually after VCS has autodisabled the group, make sure that the group is not fully or partially active on any system that has the `AutoDisabled` attribute set to 1 by VCS. Specifically, verify that all resources that may be corrupted by being active on multiple systems are brought down on the designated systems. Then, clear the `AutoDisabled` attribute for each system:

```
# hagrp -autoenable service_group -sys system
```



Service Group is Frozen

Use the output of the command `hagrp -display service_group` to verify the value of the Frozen and TFrozen attributes. Use the command `hagrp -unfreeze` to thaw the group.

Failover Service Group is Online on Another System

The group is a failover group and is online or partially online on another system. Use the output of the command `hagrp -display service_group` to verify the value of the State attribute. Use the command `hagrp -offline` to offline the group on another system.

Service Group Waiting for Resource to be Brought Online

Review the IState attribute of all resources in the service group to locate which resource is waiting to go online. Use the `hastatus` command to help identify the resource. See the engine and agent logs for information on why the resource is unable to be brought online.

- ◆ On UNIX: `/var/VRTSvcs/log`
- ◆ On Windows NT: `%VCS_HOME%\log`

To clear this state, make sure all resources waiting to go online do not bring themselves online. Use the command `hagrp -flush` to clear the internal state of VCS. You can now bring the service group online on another system.

Critical Resource Has Faulted

Output of the command `hagrp -display service_group` indicates that the service group has faulted. Use the command `hares -clear` to clear the fault.

Service Group Waiting for Dependency to be Satisfied

To see which dependencies have not been met, review the information at the bottom of the output of `hagrp -display service_group`.

Service Group Not Fully Probed

This occurs if the agent processes have not monitored each resource in the service group. Use the output of `hagrp -display service_group` to see the value of the `ProbesPending` attribute for the system's service group. (It should be zero.)

When the VCS engine starts, it immediately “probes” to find the initial state of all of resources. (It cannot probe if the agent is not returning a value.) To determine which resources are not probed, verify the local `Probed` attribute for each resource on the specified system. Zero means waiting for probe result, 1 means probed, and 2 means VCS not booted. See the engine and agent logs for more information:

- ◆ On **UNIX**: `/var/VRTSvcs/log`
- ◆ On **Windows NT**: `%VCS_HOME%\log`

A service group must be probed on all systems included in the `SystemList` attribute before VCS attempts to bring the group online as part of `AutoStart`. This ensures that even if the service group was online prior to VCS being brought up, VCS will not inadvertently bring the service group online on another system.

Bringing Resources Online

Occasionally, you may encounter problems when trying to bring system resources online. The following information describes some of these problems and provides solutions.

Waiting for Service Group States

The state of the service group prevents VCS from bringing the resource online. See page 184 for more information.

Waiting for Child Resources

One or more child resources of resource are offline. Bring the child resources online first.

Waiting for Resource to Respond

The resource is waiting to come online. Verify its `IState` attribute. VCS had directed the agent to run an online entry point for the resource. See the engine and agent logs for information on why the resource is unable to be brought online.

- ◆ On **UNIX**: `/var/VRTSvcs/log`
- ◆ On **Windows NT**: `%VCS_HOME%\log`



Agent Not Running

The resource's agent process is not running. Use `hastatus -summary` to see if the agent is listed as faulted. Restart the agent:

```
# haagent -start resource_type -sys system
```

Invalid Agent Argument List

Verify that the arguments to the scripts are correct. Use the output of `hares -display resource` to see the value of the `ArgListValues` attribute. If the `ArgList` attribute was dynamically changed, kill the agent and restart it.

Taking Service Groups Offline

Occasionally, you may encounter problems when trying to take VCS service groups offline. The following information describes some of these problems and provides solutions.

Service Group is Frozen

Use the output of `hagrps -display service_group` to see the values of `Frozen` and `TFrozen` attributes. VCS will not offline a frozen group.

Service Group is Waiting for Resource to Go Offline

The service group is waiting for an `OnOff` resource to go offline. Look at the `IState` attribute of all resources in the service group to find resources that are waiting to go offline, or use the command `hastatus`. See the engine and agent logs for information on why the resource is unable to be taken offline.

- ◆ On **UNIX**: `/var/VRTSvcS/log`
- ◆ On **Windows NT**: `%VCS_HOME%\log`

Note that you can manually offline the resource.

Taking Resources Offline

Waiting for Parent Resource

One or more parent resources are online. Take the parent resources offline first.

Waiting for Resource to Respond

The resource is waiting to go offline. Look at its `IState` attribute. See the engine and agent logs for information on why the resource is unable to be taken offline.

- ◆ On **UNIX**: `/var/VRTSvcs/log`
- ◆ On **Windows NT**: `%VCS_HOME%\log`

Agent Not Running

The resource's agent process is not running. Use `hastatus -summary` to see if the agent is listed as faulted. Restart the agent:

```
# haagent -start resource_type -sys system
```

Invalid Agent Argument List

Verify that the arguments to the scripts are correct. Use the output of `hares -display resource` to see the value of the `ArgListValues` attribute. If the `ArgList` attribute was dynamically changed, kill the agent and restart it.

Stopping VCS Without -force Option

When VCS is stopped on a system without using the `-force` option to `hastop`, it enters the `LEAVING` state, and waits for all groups to go offline on the system. Use the output of the command `hasys -display system` to verify that the values of the `SysState` and the `OnGrpCnt` attributes are non-zero. VCS continues to wait for the service groups to go offline before it shuts down. See [“Taking Service Groups Offline”](#) on page 188 for more information.



Connecting to Cluster Manager from the Command Line

Occasionally, you may encounter a problem when trying to log in to Cluster Manager from the command line. If after entering your name and password you receive the message “Cannot connect, please try again later,” wait several seconds for the VCS engine to come up and try logging in again. The list of systems to which the Cluster Manager attempts to connect on is contained in the following files:

- ◆ On **UNIX**: `$VCS_HOME/gui/conf/VCS_Sample.properties`
- ◆ On **Windows NT**: Double-click the Cluster Manager icon. From the File menu select New Cluster. Enter the required information in the New Cluster-Connectivity Configuration dialog box and click OK.

By default, the only system listed is `localhost`. Add other systems to the list by manually incrementing `ipm.NumOfNodes` and adding the following lines (the variable `num` monotonically increases beginning with 2):

```
ipm.HOST.num=system
ipm.port.num=14141
```

When you have completed adding systems, kill Cluster Manager and restart.

Primary Domain Controller File

If the syntax of the primary domain controller (PDC) file has been modified incorrectly or deleted, you will encounter problems when adding a system to the cluster or when uninstalling VCS. The following example represents the correct syntax of the VCS PDC file on a two-system cluster:

```
ClusterName      VCSClus 5
NODE: THORNT32   ID:1
NODE: THORNT33   ID:2
```

The actual location of the Primary Domain Controller file is `C:\Program Files\Veritas\VCS_PDC\VCSPDC.conf`. This file is located on the PDC and is mandatory. Note that the installation process never mentions that the file is being created, or where it is located.

The file is a text file containing information regarding each VCS cluster created on that domain. It includes the cluster name and LLT cluster ID for each cluster. Additionally, it lists each system in the cluster and the associated LLT node ID.

During addition and removal of clusters and cluster members in a domain, the PDC file is queried by the installation program. Based on existing values, the installation program automatically chooses LLT cluster and node IDs. The installation program displays the values, which are chosen automatically, however the user does not have the option of changing them. The new values are then updated in the PDC file.

Typically, this is a convenient method for the installer. However, if something happens to the PDC file, the installation program may return an invalid configuration. This can occur a number of ways; for example, if the PDC was rebuilt, or if the PDC was demoted to a Backup Domain Controller (BDC). It may also occur if an administrator inadvertently removed the file. Regardless of the scenario, if the PDC file is modified incorrectly, the addition of new clusters and systems may result in a LLT ID collision. These collisions may also occur if you have multiple clusters sharing heartbeat hubs and the clusters are in different domains. In this scenario cluster and node IDs are *not* coordinated, which increases the risk of a collision.

If you suspect that the LLT IDs automatically assigned by the installation program may be invalid, you must resolve them before rebooting. When the installation program asks if it should automatically reboot the member systems, enter No. At that point, you can manually change the LLT IDs before rebooting the systems.

In any case, verify that the IDs assigned by you or the installation program do not collide with any VCS clusters that are sharing heartbeat networks. This especially important if you are doing a `link-lowpri` on the public network.

SNMP Traps

VCS sends SNMP V1 traps to the Network Management Console when certain events occur, such as a system joining or leaving a cluster or a group starting on a cluster.

There are seven traps defined in VCS, and these traps can be enabled by configuring the required attributes in the VCS configuration file, `main.cf`: `IPAddr` and `Port`. Set the `IPAddr` attribute to configure the IP address of the Network Management Console. Change the default port value (162) by redefining the `Port` attribute. These attributes are described on page 224. Note that traps sent to the Network Management Console do not resolve to a specific enterprise, but the message generated by the trap is displayed.

Sample Trap Messages

Trap messages are strings containing information about events that have occurred in VCS. Trap messages are sent as TRAP protocol data units (PDUs).

Sample trap messages include:

```
2000-01-26 12:16:50 thor19.veritas.com [166.98.16.117]
  enterprises.3.1.1:
    Enterprise Specific Trap (2) Uptime: 109 days, 19:52:51
    enterprises = "Node named: thor19 changed state from:
    LOCAL_BUILD to: RUNNING."

2000-01-26 12:16:50 thor19.veritas.com [166.98.16.117]
  enterprises.3.1.1:
    Enterprise Specific Trap (1) Uptime: 109 days, 19:52:51
```



```
enterprises = "System thor19 joined the cluster."
2000-01-26 12:16:50 thor19.veritas.com [166.98.16.117]
enterprises.3.1.1:
  Enterprise Specific Trap (3) Uptime: 109 days, 19:52:51
  enterprises = "Group Diamond1 has been probed on system thor19."
2000-01-26 12:16:50 thor19.veritas.com [166.98.16.117]
enterprises.3.1.1:
  Enterprise Specific Trap (3) Uptime: 109 days, 19:52:51
  enterprises = "Group Diamond1 is online on system thor19"
```

When VCS Shuts Down a System

In some cases, VCS kernel components may intentionally bring down a system to avoid network partitioning. See the *VERITAS Cluster Server Release Notes* for details.

System States

B

Whenever the VCS engine is running on a system, it is in one of the states described in the table below. States indicate a system's current mode of operation. When the engine is started on a new system, it identifies the other systems available in the cluster and their states of operation. If a cluster system is in the state of `RUNNING`, the new system retrieves the configuration information from that system. Changes made to the configuration while it is being retrieved are applied to the new system before it enters the `RUNNING` state.

If no other systems are up and in the state of `RUNNING` or `ADMIN_WAIT`, and the new system has a configuration that is not marked "stale," the engine transitions to the state `LOCAL_BUILD`, and builds the configuration from disk. If the configuration is marked "stale," the system transitions to the state of `STALE_ADMIN_WAIT`.

The following table provides a list of VCS system states and their descriptions. Examples of how states transition from one state to another begin on page 195.

State	Definition
<code>ADMIN_WAIT</code>	The running configuration was lost.
<code>CURRENT_DISCOVER_WAIT</code>	The system has joined the cluster and its configuration file is valid. The system is waiting for information from other systems before it determines how to transition to another state.
<code>CURRENT_PEER_WAIT</code>	The system has a valid configuration file and another system is doing a build from disk (<code>LOCAL_BUILD</code>). When its peer finishes the build, this system transitions to the state <code>REMOTE_BUILD</code> .
<code>EXITING</code>	The system is leaving the cluster.
<code>EXITED</code>	The system has left the cluster.
<code>EXITING_FORCIBLY</code>	An <code>hastop -force</code> command has forced the system to leave the cluster.



State	Definition
FAULTED	The system has left the cluster unexpectedly.
INITING	The system has joined the cluster. This is the initial state for all systems.
LEAVING	The system is leaving the cluster gracefully. When the agents have been stopped, and when the current configuration is written to disk, the system transitions to EXITING.
LOCAL_BUILD	The system is building the running configuration from the disk configuration.
REMOTE_BUILD	The system is building a running configuration that it obtained from a peer in a RUNNING state.
RUNNING	The system is an active member of the cluster.
STALE_ADMIN_WAIT	The system has a stale configuration and there is no other system in the state of RUNNING from which to retrieve a configuration. If a system with a valid configuration is started, that system enters the LOCAL_BUILD state. Systems in STALE_ADMIN_WAIT transition to STALE_PEER_WAIT.
STALE_DISCOVER_WAIT	The system has joined the cluster with a stale configuration file. It is waiting for information from any of its peers before determining how to transition to another state.
STALE_PEER_WAIT	The system has a stale configuration file and another system is doing a build from disk (LOCAL_BUILD). When its peer finishes the build, this system transitions to the state REMOTE_BUILD.
UNKNOWN	The system has not joined the cluster because it does not have a system entry in the configuration.

Examples of State Transitions

- ◆ If VCS is started on a system, and if that system is the only one in the cluster with a valid configuration, the system transitions to the `RUNNING` state:

`INITING` → `CURRENT_DISCOVER_WAIT` → `LOCAL_BUILD` → `RUNNING`

- ◆ If VCS is started on a system with a valid configuration file, and if at least one other system is already in the `RUNNING` state, the new system transitions to the `RUNNING` state:

`INITING` → `CURRENT_DISCOVER_WAIT` → `REMOTE_BUILD` → `RUNNING`

- ◆ If VCS is started on a system with a stale configuration file, and if at least one other system is already in the `RUNNING` state, the new system transitions to the `RUNNING` state:

`INITING` → `STALE_DISCOVER_WAIT` → `REMOTE_BUILD` → `RUNNING`

- ◆ If VCS is started on a system with a stale configuration file, and if all other systems are in `STALE_ADMIN_WAIT` state, the system transitions to the `STALE_ADMIN_WAIT` state as shown below. A system stays in this state until another system with a valid configuration file is started, or when the command `hasys -force` is issued. (See page 55.)

`INITING` → `STALE_DISCOVER_WAIT` → `STALE_ADMIN_WAIT`

- ◆ If VCS is started on a system with a valid configuration file, and if other systems are in the `ADMIN_WAIT` state, the new system transitions to the `ADMIN_WAIT` state. (For information on how to exit this state, see the `hasys` command on page 55.)

`INITING` → `CURRENT_DISCOVER_WAIT` → `ADMIN_WAIT`

- ◆ If VCS is started on a system with a stale configuration file, and if other systems are in the `ADMIN_WAIT` state, the new system transitions to the `ADMIN_WAIT` state. (For information on how to exit this state, see the `hasys` command on page 55.)

`INITING` → `STALE_DISCOVER_WAIT` → `ADMIN_WAIT`

- ◆ When a system in `RUNNING` state is stopped with the `hastop` command, it transitions to the `EXITED` state as shown below. During the `LEAVING` state, any online system resources are taken offline. When all of the system's resources are taken offline and the agents are stopped, the system transitions to the `EXITING` state.

`RUNNING` → `LEAVING` → `EXITING` → `EXITED`





VCS Attributes



Resource Attributes

Attributes	Type, Dimension, Scope	Definition
ArgListValues	string-vector local	Argument vector passed to the resource's agent on each system.
AutoStart	boolean-scalar global	Indicates that the resource is brought online when the service group is brought online. Default = 1
ConfidenceLevel	integer-scalar local	Indicates the level of confidence in an online resource. Values range from 0-100. Note that some VCS agents may not take advantage of this attribute and may always set it to 0. Set the level to 100 if the attribute is not used.
Critical	boolean-scalar global	Indicates that the service group is faulted when the resource, or any resource it depends on, faults. Default = 1
Enabled	boolean-scalar global	Indicates that agents monitor this resource. If you specify the resource in main.cf prior to starting VCS, the default value for this attribute is 1. When a resource is created dynamically when VCS is running, you must enable the resource before VCS will monitor it. (For more information on how to add or enable resources, see XXX (command line and gui chapter.)



Attributes	Type, Dimension, Scope	Definition
Flags	integer, scalar local	<p>Additional information relating to the state of a resource.</p> <p>Values:</p> <p>RESTARTING indicates that the resource faulted and that the agent is attempting to restart the resource on the same system.</p> <p>STATE UNKNOWN indicates that the latest monitor call by the agent could not determine if the resource is online or offline.</p> <p>MONITOR TIMEDOUT indicates that the latest monitor call by the agent was terminated because it exceeded the maximum time specified by the static attribute MonitorTimeout.</p> <p>UNABLE TO OFFLINE indicates that the agent attempted to offline the resource but the resource is not going offline. This flag is also set when a resource faults and clean completes successfully, but the subsequent monitor hangs or is unable to determine resource status.</p>
Group	string-scalar global	String name of the group.
LastOnline	string-scalar global	Indicates the system name on which the resource was last online. This attribute is automatically set by the VCS engine (HAD).
MonitorOnly	boolean-scalar global	Indicates if the resource can be brought online or taken offline.

Attributes	Type, Dimension, Scope	Definition
IState	integer-scalar local	<p>Internal state of a resource.</p> <p>Values:</p> <p>NOT WAITING Resource is not in transition.</p> <p>WAITING TO GO ONLINE Agent notified to bring the resource online but procedure not yet complete.</p> <p>WAITING FOR CHILDREN ONLINE Resource to be brought online, but resource depends on at least one offline resource. Resource transitions automatically to WAITING TO GO ONLINE when all children are online.</p> <p>WAITING TO GO OFFLINE Agent notified take the resource offline but procedure not yet complete.</p> <p>WAITING TO GO OFFLINE (propagate) Same as above, but when completed the resource's children will also be offline.</p> <p>WAITING TO GO ONLINE (reverse) Resource waiting to be brought online, but when it is online it automatically attempts to go offline. Typically this is the result of issuing an offline command while resource was waiting to go online.</p> <p>WAITING TO GO OFFLINE (reverse/propagate) Same as above but resource propagates offlining.</p>
Path	boolean-scalar local	For internal use only.
Name	string-scalar global	For internal use only.
Probed	boolean-scalar local	Indicates whether the resource has been detected by the agent.



Attributes	Type, Dimension, Scope	Definition
ResourceOwner	string-scalar global	<p>Sends notifications to owners when something goes wrong with their resource.</p> <p>Resource state change messages resemble:</p> <pre> TAG_E 2000/12/03 11:23:48 VCS:10304:Resource file1 (Owner=Daniel Group testgroup) is offline on thor80. </pre> <p>If ResourceOwner is not specified in main.cf, the default value is “unknown.” For example:</p> <pre> TAG_E 2000/12/03 11:23:48 VCS:10304:Resource file1 (Owner=unknown Group testgroup) is offline on thor80 </pre>
Signaled	integer-association local	For internal use only.
Start	integer-scalar local	For internal use only.
State	integer-scalar local	<p>Resource state displays the state of the resource and the flags associated with the resource. (Flags are also captured by the Flags attribute, described on page 198.)</p> <p>Values:</p> <p>ONLINE</p> <p>OFFLINE</p> <p>FAULTED</p> <p>ONLINE STATE UNKNOWN</p> <p>ONLINE MONITOR TIMEDOUT</p> <p>ONLINE UNABLE TO OFFLINE</p> <p>OFFLINE STATE UNKNOWN</p> <p>FAULTED RESTARTING</p> <p>A FAULTED resource is physically offline, though unintentionally.</p>
TriggerEvent	integer-scalar global	For internal use only.



Type-Specific Resource Attributes

Type-specific resource attributes are declared in the type definition (types.cf). The following table lists the type-specific attributes for Mount resource type on UNIX.

Attributes	Type, Dimension, Scope	Definition
BlockDevice	string-scalar global	Block device for the mount point.
FsckOpt	string-scalar global	Options for the <code>fsck</code> command.
MountOpt	string-scalar global	Options for the <code>mount</code> command.
MountPoint	string-scalar global	Directory for the mount point.
Type	string-scalar global	File system type, such as <code>vxfs</code> , <code>ufs</code> , etc.



The following table lists the type-specific attributes for Mount resource type on Windows NT.

Attributes	Type, Dimension, Scope	Definition
DiskResName	string-scalar global	The name of the Disk resource for which the Mount resource is configured.
DriveLetter	string-scalar global	The drive letter assigned to the device path being mounted.
FileSystemType	string-scalar global	The name of the file system on the partition being mounted. This version supports NTFS and FAT.
ForceUnmount	boolean-scalar global	Defines whether the agent unmounts the volume forcefully. Default = 0
ListApplication	boolean-scalar global	<p>Defines whether the agent lists which applications are accessing the mount point while unmounting. Default= 1.</p> <p>You can define local attributes for this resource in the main.cf file. For example:</p> <pre>ListApplication@sysa = 0 ListApplication@sysb = 1</pre> <p>This designates the local value of this attribute to each system in the cluster.</p>
PartitionNo	integer-scalar global	<p>The partition on the Disk resource configured for mounting. Note that the base index for the partition number is 1, and the base index for the disk number is 0.</p> <p>You can define local attributes for this resource in the main.cf file. For example:</p> <pre>PartitionNo@sysa = 1 PartitionNo@sysb = 2</pre> <p>This designates the local value of this attribute to each system in the cluster.</p>

Additional Attributes for Scheduling Class and Priority Configuration Support

VCS allows you to specify priorities and scheduling classes for VCS processes. For more information on this feature, including information regarding default priority values, see [“Scheduling Class and Priority Configuration Support”](#) on page 165. Instructions on initializing attributes in the types.cf file or setting them from the command line in the section following the table below.

Attributes	Type, Dimension, Scope	Definition
AgentClass	string-scalar global	Indicates the scheduling class for the VCS agent process. Default = "TS"
AgentPriority	string-scalar global	Indicates the priority in which the agent process runs. Default = ""
ScriptClass	string-scalar global	Indicates the scheduling class of the script processes (for example, online) created by the agent. Default = "TS"
ScriptPriority	string-scalar global	Indicates the priority of the script processes created by the agent. Default = ""

Initializing Attributes in the Configuration File

The following configuration shows how to initialize these attributes through configuration files. The example shows attributes of a FileOnOff resource.

```

type FileOnOff (
    static str AgentClass = RT
    static str AgentPriority = 10
    static str ScriptClass = RT
    static str ScriptPriority = 40
    static str ArgList[] = { PathName }
    NameRule = resource.PathName
    str PathName
)

```



Setting Attributes Dynamically from the Command Line

▼ To update the AgentClass

Type:

```
hatype -modify resource_type AgentClass value
```

For example, to set the AgentClass attribute of the FileOnOff resource to Realtime, type:

```
hatype -modify FileOnOff AgentClass "RT"
```

▼ To update the AgentPriority

Type:

```
hatype -modify resource_type AgentPriority value
```

For example, to set the AgentPriority attribute of the FileOnOff resource to 10, type:

```
hatype -modify FileOnOff AgentPriority "10"
```

▼ To update the ScriptClass

Type:

```
hatype -modify resource_type ScriptClass value
```

For example, to set the ScriptClass of the FileOnOff resource to RealTime, type:

```
hatype -modify FileOnOff ScriptClass "RT"
```

▼ To update the ScriptPriority

Type:

```
hatype -modify resource_type ScriptPriority value
```

For example, to set the ScriptClass of the FileOnOff resource to RealTime, type:

```
hatype -modify FileOnOff ScriptPriority "40"
```

Note Note: For the attributes AgentClass and AgentPriority, changes are effective immediately. For ScriptClass and ScriptPriority, changes become effective for scripts fired after the execution of the `hatype` command.

Service Group Attributes

Attributes	Type, Dimension, Scope	Definition
Active Count	integer-scalar	Number of resources in a service group that are active (online or waiting to go online). When the number of resources drops to zero, the service group is considered offline.
AutoDisabled	boolean-scalar local	When VCS does not know the status of a service group on a particular system, it autodisables the service group on that system. See page 185 for details.
AutoFailOver	boolean-scalar global	Indicates if automatic failover is enabled for the service group. Default = 1 (enabled)
AutoStart	boolean-scalar global	Designates whether a service group will be automatically started when VCS is started. The value 1 indicates that the group will be started. The value 0 indicates that it will not. Default = 1 (enabled)
AutoStartList	string-keylist global	List of systems on which the service group will be started with VCS (usually at system boot). Note For the service group to start, AutoStart must be enabled and Frozen must be 0. Also, beginning with 1.3.0, you must define the SystemList attribute prior to setting this attribute. Default = "" (none)
CurrentCount	integer-scalar global	Number of systems on which the service groups is active.
Enabled	boolean-scalar local	Indicates if a group can be failed over or brought online. If any of the local values are disabled, the group is disabled. Default = 1 (enabled)



Attributes	Type, Dimension, Scope	Definition
AutoRestart	integer-scalar global	<p>Restarts a service group after a faulted persistent resource becomes online.</p> <p>For example, if a persistent resource on group1 faults, the service group is automatically failed over to another system in the cluster under the following conditions:</p> <ul style="list-style-type: none"> ◆ The AutoFailover attribute is set. ◆ There is another system in the cluster to which group1 can fail over. <p>If neither of the above conditions is met (the AutoFailover attribute is not set or other systems in the cluster are unavailable), group1 remains offline and faulted, even after the faulted resource becomes online.</p> <p>Setting the AutoRestart attribute enables a service group to be brought back online without manual intervention. In the above example, setting the AutoRestart attribute for the group1 would enable VCS to bring the group back online, after the resource had become online, on the system where the resource had faulted.</p> <p>Or, if group1 could not fail over to another system because none was available, setting the AutoRestart attribute would enable VCS to bring the group back online on the first available system after the group's faulted resource had become online.</p> <p>For example, NIC is a persistent resource. In some cases, when a system boots and VCS starts, VCS probes all resources on the system. It is possible that when VCS probes the NIC resource, the resource may not yet be online because the networking is not up and fully operational. When this occurs, VCS will mark the NIC resource as faulted, and will not bring the service group online. However, when the NIC resource becomes online and if AutoRestart is enabled, the service group is brought online.</p> <p>To enable this attribute, set the value to 1.</p> <p>Note This attribute applies to persistent resources only.</p>

Attributes	Type, Dimension, Scope	Definition
Evacuating	integer-scalar global	For internal use only.
ExtMonApp	string-scalar global	Not yet implemented.
ExtMonArgs	string-vector global	Not yet implemented.
Failover	boolean-scalar global	For internal use only.
FailOverPolicy	string-scalar global	<p>Sets the policy VCS uses to determine which system a group fails over to if multiple systems exist.</p> <p>Values:</p> <p>Priority (default): The system defined as the lowest priority in the SystemList attribute will be chosen.</p> <p>Load: The system defined with the least value in the system's Load attribute will be chosen.</p> <p>RoundRobin: The system with the least number of active service groups will be chosen.</p>
FromQ	string-association global	For internal use only.
Frozen	boolean-scalar global	<p>Disables all actions, including autostart, failover, and non-monitor actions performed by agents. (This convention is observed by all agents supplied with VCS.)</p> <p>Default = 0 (not frozen)</p>
IntentOnline	integer-scalar global (failover groups) local (parallel groups)	<p>This attribute designates whether to keep service groups online or offline. It is set to 1 by the VCS if an attempt has been made, successful or not, to online the service group. For failover groups, this attribute is set to 0 by VCS when the group is taken offline. For parallel groups, it is set to 0 for either system when the group is taken offline or when the group faults and can failover to another system.</p>



Attributes	Type, Dimension, Scope	Definition
LastSuccess	integer-scalar global	For internal use only.
ManualOps	boolean-scalar global	Indicates if manual operations are allowed on the service group. Default = 1
MigrateQ	string-association global	For internal use only.
NumRetries	integer-scalar global	For internal use only.
OnlineRetryInterval	integer-scalar global	Indicates the interval, in seconds, during which a service group that has successfully restarted on the same system and faults again should be failed over, even if the attribute OnlineRetryLimit is non-zero. This prevents a group from continuously faulting and restarting on the same system.
OnlineRetryLimit	integer-scalar global	If non-zero, specifies the number of times the VCS engine tries to restart a faulted service group on the same system on which the group faulted, before it gives up and tries to fail over the group to another system.
Parallel	boolean-integer global	Indicates if service group is parallel (1) or failover (0). Default = 0
PathCount	integer-scalar	Number of resources in path not yet taken offline. When this number drops to zero, the engine may take the entire service group offline if the fault is critical occurred; for example, if the Failover attribute is set to 1.
PreOffline	integer-scalar global	For internal use only.



Attributes	Type, Dimension, Scope	Definition
PreOnline	integer-scalar global	Indicates that the VCS engine should not online a service group in response to an <code>hagrp -online</code> command or a fault. The engine should instead call a user-defined script that checks for external conditions before bringing the group online. See “ Event Notification ” on page 157 for details.
PreOfflining	integer-scalar global	For internal use only.
PreOnlining	integer-scalar global	For internal use only.
Priority	integer-scalar global	Enables users to designate and prioritize the service group. VCS does not interpret the value; rather, this attribute enables the user to configure the priority of a service group and the sequence of actions required in response to a particular event.
PrintTree	boolean-scalar global	Indicates whether the resource dependency tree should be written to the configuration file.
ProbesPending	integer-scalar local	The number of resources that remain to be detected by the agent on each system.
Responding	integer-scalar global	For internal use only.
Restart	integer-scalar local	For internal use only.
SourceFile	string-scalar global	File from which the configuration was read. Always set to <code>./types.cf</code> .



Attributes	Type, Dimension, Scope	Definition
State	integer-scalar local	<p>Group state on each system:</p> <p>OFFLINE All non-persistent resources are offline.</p> <p>ONLINE All resources whose AutoStart attribute is equal to 1 are online.</p> <p>FAULTED At least one critical resource in the group is faulted or is affected by a fault.</p> <p>PARTIAL At least one, but not all, resources whose AutoStart attribute is equal to 1 are online in the group.</p> <p>STARTING Group is attempting to go online.</p> <p>STOPPING Group is attempting to go offline.</p> <p>Note It is possible that a group state is a combination of the multiple states described above. For example, OFFLINE FAULTED.</p>
SystemList	string-association global	<p>List of systems on which the service group is configured to run and their priorities. Lower numbers indicate a preference for the system as a failover target.</p> <p>Note Beginning with 1.3.0, you must define this attribute prior to setting the AutoStartList attribute.</p> <p>Default = "" (none)</p>
SystemZones	integer-association global	<p>Indicates the virtual sublists within the SystemList attribute that grant priority in failing over. Values are string/integer pairs. The string key is the name of a system in the SystemList attribute, and the integer is the number of the zone. Systems with the same zone number are members of the same zone. If a service group faults on one system in a zone, it is granted priority to fail over to another system within the same zone, despite the policy granted by the FailOverPolicy attribute (page 207).</p>
TargetCount	integer-scalar global	For internal use only.
TFrozen	boolean-scalar global	Indicates if a group can be brought online or taken offline.

Attributes	Type, Dimension, Scope	Definition
ToQ	string-association global	For internal use only.
TriggerEvent	integer-scalar global	For internal use only.
UserIntGlobal	integer-scalar global	Use this attribute for any purpose. It is not used by VCS. Default = 0
UserStrGlobal	string-scalar global	Use this attribute for any purpose. It is not used by VCS. Default = ""
TypeDependencies	string-vector global	Creates a dependency between resource types specified in the service group list, and all instances of the respective resource type.
UserIntLocal	integer-scalar local	Use this attribute for any purpose. It is not used by VCS. Default = 0
UserStrLocal	string-scalar local	Use this attribute for any purpose. It is not used by VCS. Default = ""



System Attributes

Attributes	Type, Dimension, Scope	Definition
AgentsStopped	integer-scalar global	For internal use only.
ConfigBlockCount	integer-scalar global	Number of 512-byte blocks in configuration when the system joined the cluster.
ConfigChecksum	integer-scalar global	Sixteen-bit checksum of configuration identifying when the system joined the cluster.
ConfigDiskState	integer-scalar global	State of configuration on the disk when the system joined the cluster.
ConfigFile	string-scalar global	Directory containing the configuration files.
ConfigInfoCnt	integer-scalar global	For internal use only.
ConfigModDate	integer-scalar global	Last modification date of configuration when the system joined the cluster.
DiskHbDown	string-vector global	Indicates if communication disks are down on any system. Enabled by the LinkMonitoring attribute.
Frozen	boolean-scalar global	Indicates if service groups can be brought online on the system. Groups cannot be brought online if the attribute value is 1.
GUIIPAddr	string-scalar global	Determines the local IP address that VCS will use to accept connections. Incoming connections over other IP addresses are dropped. If GUIIPAddr is not set, the default behavior is to accept external connections over all configured local IP addresses. For more information on this attribute, see “VCS Security” on page 166.



Attributes	Type, Dimension, Scope	Definition
LinkHbDown	string-vector global	Indicates if private network links are down on any system. Enabled by the LinkMonitoring attribute.
LLTNodeID	integer-scalar global	For internal use only.
Load	integer-scalar global	Normalized value of system load used to compare systems in load balancing. Value is determined by dividing the raw values of LoadRaw (described below) by the values of Factor.
LoadRaw	integer-association global	List of load-calculation criterion and their associated raw values over the last five seconds. These criteria are the number of processes waiting to run (runque), the number of times page daemon has run (memory), the number of disk writes (disk), CPU utilization percentage (cpu), and network traffic, not including GAB and LLT (network).
MajorVersion	integer-scalar global	Major version of system's join protocol.
MinorVersion	integer-scalar global	Minor version of system's join protocol.
NodeId	integer-scalar global	System ID specified in /etc/llttab.
OnGrpCnt	integer-scalar global	Number of groups that are online, or about to go online.



Attributes	Type, Dimension, Scope	Definition
ShutdownTimeout	integer-scalar global	<p>Determines whether to treat system reboot as a fault for service groups running on the system.</p> <p>On many systems, when a reboot occurs the processes are killed first, then the system goes down. When the VCS engine is killed, service groups that include the failed system in their SystemList attributes are autodisabled. However, if the system goes down within the number of seconds designated in ShutdownTimeout, service groups previously online on the failed system are treated as faulted and failed over.</p> <p>If you do not want to treat the system reboot as a fault, set the value for this attribute to 0.</p> <p>Default = 60 seconds</p> <p>VCS uses the GlobalCounter attribute to measure the time it takes to shut down a system. By default, the GlobalCounter attribute is updated every five seconds (see page 220). This default value, combined with the 60-second default value of ShutdownTimeout, means that if system goes down within twelve increments of GlobalCounter, it will be treated as a fault. The default value of GlobalCounter increment can be modified by changing the CounterInterval attribute (page 220). If you increase the CounterInterval attribute to exceed five seconds, consider increasing the default value of the ShutdownTimeout attribute as well.</p>
SourceFile	string-scalar global	File from which the configuration was read. Always set to ./main.cf.
SysInfo	string-scalar global	<p>Provides platform-specific information, including the name, version, and release of the operating system, the name of the system on which it is running, and the hardware type. On Windows NT, this information also includes the build number, service pack, processor, and number of processors.</p>

Attributes	Type, Dimension, Scope	Definition
SysState	integer-scalar global	System state. (See “ System States ” on page 193 for more information.)
TFrozen	boolean-scalar global	Indicates if a group can be brought online or taken offline.
TRSE	integer-scalar global	For internal use only.
UpDownState	integer-scalar global	<p>This attribute has four values:</p> <p>DOWN: System is powered off, or GAB and LLT are not running on the system.</p> <p>UP BUT NOT IN CLUSTER MEMBERSHIP:</p> <ul style="list-style-type: none"> ◆ GAB and LLT are running but the VCS engine is not. ◆ The system is recognized through disk heartbeat only. <p>UP AND IN JEOPARDY: The system is up and part of cluster membership, but only one network link (LLT) remains.</p> <p>UP: The system is up and part of cluster membership, and has at least two links to the cluster.</p> <p>For more information, see “About Cluster Memberships” on page 167.</p>
UserInt	integer-scalar global	Stores a system’s integer value.
UserStr	integer-scalar global	Stores a system’s string value.



Resource Type Attributes

For more information on any attribute listed below, see the chapter on setting agent parameters in the *VERITAS Cluster Server Agent Developer's Guide*.

Attributes	Type, Dimension, Scope	Definition
AgentClass	string-scalar	Indicates the scheduling class for the VCS agent process. (See page 203 for more information on how to use this attribute to schedule class and priority configuration support.) Default = "TS"
AgentFailedOn	string-vector global	A list of systems on which the agent for the resource type has failed.
AgentPriority	string-scalar	Indicates the priority in which the agent process runs. (See page 203 for more information on how to use this attribute to schedule class and priority configuration support.) Default = ""
AgentReplyTimeout	integer-scalar global	The number of seconds the engine waits to receive a heartbeat from the agent before restarting the agent. Default = 130 seconds
AgentStartTimeout	integer-scalar global	The number of seconds after starting the agent that the engine waits for the initial agent "handshake" before restarting the agent. Default = 60 seconds
ArgList	string-vector global	An ordered list of attributes whose values are passed to the <code>open</code> , <code>close</code> , <code>online</code> , <code>offline</code> , <code>monitor</code> , and <code>clean</code> entry points. Default = ""
AttrChangedTimeout	integer-scalar global	Maximum time (in seconds) within which the <code>attr_changed</code> entry point must complete or be terminated. Default = 60 seconds

Attributes	Type, Dimension, Scope	Definition
CleanTimeout	integer-scalar global	Maximum time (in seconds) within which the <code>clean</code> entry point must complete or else be terminated. Default = 60 seconds
CloseTimeout	integer-scalar global	Maximum time (in seconds) within which the <code>close</code> entry point must complete or else be terminated. Default = 60 seconds
ConfInterval	integer-scalar global	When a resource has remained online for the specified time (in seconds), previous faults and restart attempts are ignored by the agent. (See <code>ToleranceLimit</code> and <code>RestartLimit</code> attributes for details.) Default = 600 seconds
FaultOnMonitorTimeouts	integer-scalar global	When a monitor fails as many times as the value specified, the corresponding resource is brought down by calling the <code>clean</code> entry point. The resource is then marked <code>FAULTED</code> , or it is restarted, depending on the value set in the <code>Restart Limit</code> attribute. When <code>FaultOnMonitorTimeouts</code> is set to 0, monitor failures are not considered indicative of a resource fault. Default = 4
LogLevel	string-scalar global	Specifies the type of messages to be logged to the system's local log file.
MonitorInterval	integer-scalar global	Duration (in seconds) between two consecutive monitor calls for an <code>ONLINE</code> or transitioning resource. Default = 60 seconds
MonitorTimeout	integer-scalar global	Maximum time (in seconds) within which the <code>monitor</code> entry point must complete or else be terminated. Default = 60 seconds



Attributes	Type, Dimension, Scope	Definition
OfflineMonitorInterval	integer-scalar global	Duration (in seconds) between two consecutive monitor calls for an OFFLINE resource. If set to 0, OFFLINE resources are not monitored. Default = 300 seconds
NameRule	string-scalar global	Generates the unique name of the resource.
NumThreads	integer-scalar global	Number of threads used within the agent process for managing resources. This number does not include the three threads used for other internal purposes. Default = 10
OfflineTimeout	integer-scalar global	Maximum time (in seconds) within which the <code>offline</code> entry point must complete or else be terminated. Default = 300 seconds
OnlineRetryLimit	integer-scalar global	Number of times to retry <code>online</code> , if the attempt to online a resource is unsuccessful. This parameter is meaningful only if <code>clean</code> is implemented. Default = 0
OnlineTimeout	integer-scalar global	Maximum time (in seconds) within which the <code>online</code> entry point must complete or else be terminated. Default = 300 seconds
OnlineWaitLimit	integer-scalar global	Number of monitor intervals to wait after completing the online procedure, and before the resource becomes online. Default = 2
OpenTimeout	integer-scalar global	Maximum time (in seconds) within which the <code>open</code> entry point must complete or else be terminated. Default = 60 seconds

Attributes	Type, Dimension, Scope	Definition
Operations	string-scalar global	Indicates valid operations of resources of the resource type. Values are OnOnly (can online only), OnOff (can online and offline), None (cannot online or offline). Default = OnOff
RestartLimit	integer-scalar global	Affects how the agent responds to a resource fault. Default = 0
ScriptClass	string-scalar	Indicates the scheduling class of the script processes (for example, online) created by the agent. (See page 203 for more information on how to use this attribute to schedule class and priority configuration support.) Default = "TS"
ScriptPriority	string-scalar	Indicates the priority of the script processes created by the agent. (See page 203 for more information on how to use this attribute to schedule class and priority configuration support.) Default = ""
SourceFile	string-scalar global	File from which the configuration was read. Always set to ./main.cf.
ToleranceLimit	integer-scalar global	A non-zero ToleranceLimit allows the monitor entry point to return OFFLINE several times before the resource is declared FAULTED. Default = 0



Cluster Attributes

Attributes	Type, Dimension, Scope	Definition
ClusterName	string-scalar global	Name of cluster.
CompareRSM	integer-scalar global	For internal use only.
CounterInterval	integer-scalar global	Intervals counted by the attribute GlobalCounter.
DumpingMembership	integer-scalar global	Indicates that the engine is writing to disk.
Factor	integer-association global	For internal use only.
GlobalCounter	integer-scalar global	This counter increases incrementally by one for each counter interval.
GroupLimit	integer-scalar global	Maximum number of service groups. Default = 200
LinkMonitoring	boolean-integer global	Enables link monitoring. Default = 0
LoadSampling	boolean-integer global	For internal use only.
LogSize	integer-scalar global	Size of log file. Default value = 32MB Minimum value = 64KB Maximum value = 128MB
MajorVersion	integer-scalar global	Major version of system's join protocol.
MaxFactor	integer-association global	For internal use only.

Attributes	Type, Dimension, Scope	Definition
MinorVersion	integer-scalar global	Minor version of system's join protocol.
PrintMsg	integer-scalar global	For internal use only.
ReadOnly	integer-scalar global	Indicates that cluster is in read-only mode. (See page 63 for details.)
ResourceLimit	integer-scalar global	Maximum number of resources. Default = 5000
SourceFile	string-scalar global	File from which the configuration was read. Always set to ./main.cf.
TypeLimit	integer-scalar global	Maximum number of resource types. Default = 100
UserNames	string-association global	List of valid GUI user names and their encrypted passwords.



Additional Attributes for Scheduling Class and Priority Configuration Support

VCS allows you to specify priorities and scheduling classes for VCS processes. For more information on this feature, including information regarding default priority values, see [“Scheduling Class and Priority Configuration Support”](#) on page 165. Instructions on initializing attributes in the types.cf file or setting them from the command line in the section following the table below.

Attributes	Type, Dimension, Scope	Definition
EngineClass	string-scalar global	Indicates the scheduling class for the VCS engine (HAD). Default = "RT"
EnginePriority	string-scalar global	Indicates the priority in which HAD runs.
ProcessClass	string-scalar global	Indicates the scheduling class for the processes (for example, halink) created by HAD. Default = "TS"
ProcessPriority	string-scalar global	Indicates the priority of the processes created by HAD. Default = ""

Initializing Attributes in the Configuration File

You may assign values for the above cluster attributes while configuring the cluster. Review the following sample configuration:

```
cluster vcs-india (
  EngineClass = "RT"
  EnginePriority = "20"
  ProcessClass = "TS"
  ProcessPriority = "40"
)
```

Setting Attributes Dynamically from the Command Line

▼ To update the EngineClass

Type:

```
haclus -modify EngineClass value
```

For example, to set the EngineClass attribute to Realtime, type:

```
haclus -modify EngineClass "RT"
```

▼ To update the EnginePriority

Type:

```
haclus -modify EnginePriority value
```

For example, to set the EnginePriority to 20, type:

```
haclus -modify EnginePriority "20"
```

▼ To update the ProcessClass

Type:

```
haclus -modify ProcessClass value
```

For example, to set the ProcessClass to TimeSharing, type:

```
haclus -modify ProcessClass "TS"
```

▼ To update the ProcessPriority

Type:

```
haclus -modify ProcessPriority value
```

For example, to set the ProcessPriority to 40, type:

```
haclus -modify ProcessPriority "40"
```

Note For the attributes EngineClass and EnginePriority, changes are effective immediately. For ProcessClass and ProcessPriority changes become effective only for processes fired *after* the execution of the `haclus` command.



SNMP Attributes

Predefined Attributes	Type and Dimension	Definition
Enabled	boolean-scalar	Indicates if SNMP traps are enabled. Default = 0
IPAddr	string-scalar	IP address of the host where the SNMP Manager resides.
Port	integer-scalar	Port of SNMP server.
SnmpName	string-scalar	
SourceFile	string-scalar	
TrapList	string-association	List of traps and their descriptions. Default value: TrapList = { 1 = A new system has joined a VCS cluster. 2 = An existing system has changed its state. 3 = A service group has changed its state. 4 = One or more heartbeat links has gone down. 5 = An HA service has been manually restarted. 6 = An HA service has been manually idled. 7 = An HA service has been successfully failed over to a backup system.}

Index

A

- AgentClass attribute 203, 216
- AgentFailedOn attribute 216
- AgentPriority attribute 203, 216
- AgentReplyTimeout attribute 216
- Agents
 - definition 7
 - start/stop manually 62
- AgentStartTimeout 216
- ArgList attribute 216
- ArgListValues attribute 197
- Attributes
 - Istate 199
- AttrChangedTimeout attribute 216
- Attribute
 - IPAddr 224
 - SysState 215
- Attributes
 - AgentClass 203, 216
 - AgentFailedOn 216
 - AgentPriority 203, 216
 - AgentReplyTimeout 216
 - AgentStartTimeout 216
 - ArgList 216
 - ArgListValues 197
 - assigning definition and value 18
 - AttrChangedTimeout 216
 - AutoDisabled 205
 - AutoFailover 205
 - AutoRestart 206
 - AutoStart 197, 205
 - AutoStartList 205
 - BlockDevice 201
 - CleanTimeout 217
 - CloseTimeout 217
 - ClusterName 220
 - ConfidenceLevel 197
 - ConfigBlockCount 212
 - ConfigChecksum 212
 - ConfigDiskState 212
 - ConfigFile 212
 - ConfigInfoCnt 212
 - ConfigModDate 212
 - ConfInterval 217
 - CounterInterval 220
 - Critical 197
 - DiskHbDown 212
 - DiskResName 202
 - DriveLetter 202
 - DumpingMembership 220
 - Enabled 197, 205, 224
 - EngineClass 222
 - EnginePriority 222
 - FailOverPolicy 207
 - FaultOnMonitorTimeouts 217
 - FileSystemType 202
 - Flags 198
 - ForceUnmount 202
 - Frozen 207, 212
 - FsckOpt 201
 - GlobalCounter 220
 - Group 198
 - GroupLimit 220
 - IntentOnline 207
 - LastOnline 198
 - LinkHbDown 213
 - LinkMonitoring 220
 - ListApplication 202
 - Load 213
 - LoadRaw 213
 - LogLevel 217
 - LogSize 220
 - MajorVersion 213, 220
 - ManualOps 208
 - MaxFactor 220
 - MinorVersion 213, 221



MonitorInterval 217
MonitorOnly 198
MonitorTimeout 217
MountOpt 201
MountPoint 201
NameRule 218
NodeId 213
NumThreads 218
OfflineTimeout 218
OnGrpCnt 213
OnlineRetryInterval 208
OnlineRetryLimit 208, 218
OnlineTimeout 218
OnlineWaitLimit 218
OpenTimeout 218
Operations 219
Parallel 208
PartitionNo 202
Port 224
PreOnline 209
PrintMsg 221
PrintTree 209
Probed 199
ProbesPending 209
ProcessClass 222
ProcessPriority 222
ReadOnly 221
resource attributes 27
ResourceLimit 221
RestartLimit 219
ScriptClass 203, 219
ScriptPriority 203, 219
service group attributes 26
SourceFile 209, 214, 219, 221
State 200, 210
SystemList 210
SystemZones 210
TFrozen 210, 215
ToleranceLimit 219
TrapList 224
Type 201
TypeDependencies 211
TypeLimit 221
UserInt 215
UserIntGlobal 211
UserIntLocal 211
UserNames 221
UserStr 215
UserStrGlobal 211

UserStrLocal 211
valid data types 19
valid dimensions 19
AutoDisabled attribute 205
AutoFailover attribute 205
AutoRestart attribute 206
AutoStart attribute 197, 205
AutoStartList attribute 205

B

BlockDevice attribute 201

C

CleanTimeout attribute 217
CloseTimeout 217
Cluster
4-system configuration 2
definition 2
querying for faults 58
seeding 184
ClusterName attribute 220
Commands
haattr -add 67
haattr -add -static 67
haattr -default 67
haattr -delete 67
haattr -delete -static 67
haclus -display 56
haclus -enable 56
haclus -help 56
haclus -value 56
haconf -dump -makero 50
haconf -makerw 50
hagrp -add 68
hagrp -modify 68
hagrp -offline 59
hagrp -resources 54
hagrp -unfreeze 59
hares -clear 62
hares -dep 53, 54
hares -display 53, 54
hares -list 52, 54, 55
hares local 63
hares -modify 72
hares -offline 61
hares -offprop 61
hares -online 61
hares -probe 62
hastart 48
hastart -stale -force 48



- hastatus 57
 - hastatus -group 57
 - hastop -all 49
 - hastop -local 49
 - hastop -sys 49
 - hatype -add 66
 - hatype -delete 66
 - hatype -modify 66
 - hauser -delete 52
 - hauser -modify 51
 - Communication channels
 - definition 14
 - setting up 14
 - ConfidenceLevel attribute 197
 - ConfigBlockCount attribute 212
 - ConfigChecksum attribute 212
 - ConfigDiskState attribute 212
 - ConfigFile attribute 212
 - ConfigInfoCnt attribute 212
 - ConfigModDate attribute 212
 - ConfigInterval attribute 217
 - Console-abort sequence 183
 - CounterInterval attribute 220
 - Critical attribute 197
- D**
- Dependencies
 - between resources 27
 - configuring for service groups 144
 - definition of parent/child 8
 - offline service group 131
 - online service group 131
 - removing between resources 74
 - removing between service groups 74
 - DiskHbDown attribute 212
 - DiskResName attribute 202
 - Domain-qualified system names
 - changing 24
 - reconfiguring 24
 - DriveLetter attribute 202
 - DumpingMembership attribute 220
- E**
- Enabled attribute 197, 205, 224
 - EngineClass attribute 222
 - EnginePriority attribute 222
 - Environment variables 45
 - VCS_CONF 45
 - VCS_GAB_PORT 45
 - VCS_GAB_TIMEOUT 45
 - VCS_HAD_RESTART_TIMEOUT 46
 - VCS_HOME 45
 - VCS_LOG 46
 - VCS_PORT 46
 - VCS_TEMP_DIR 46
- F**
- Failover group 6
 - FailOverPolicy attribute 207
 - FaultOnMonitorTimeouts attribute 217
 - FileSystemType attribute 202
 - Flags attribute 198
 - ForceUnmount attribute 202
 - Frozen attribute 207, 212
 - FsckOpt attribute 201
- G**
- GAB
 - configuring 15
 - heartbeat disk regions 15
 - GlobalCounter attribute 220
 - Group attribute 198
 - GroupLimit attribute 220
 - GUI 3
 - Cluster Explorer window 84
 - Cluster Monitor window 83
 - Command Center window 86
 - establishing user account for superuser 80
 - establishing user account for VCSGuest 79
 - establishing user account for VCSOperator 79
 - Log Desk window 85
 - panels 83
 - setting up display 78
 - starting Cluster Manager 80
 - GUI commands
 - logging in to a cluster 99
 - logging out of cluster 99
- H**
- haattr -add command 67
 - haattr -add -static command 67
 - haattr -default command 67
 - haattr -delete command 67
 - haattr -delete -static command 67
 - hacf 3
 - creating multiple .cf files 29
 - definition 28



- dumping a configuration 29
 - loading a configuration 29
 - pretty-printing 28
- haclus -display command 56
- haclus -enable command 56
- haclus -help command 56
- haclus -value command 56
- haconf -dump -makero command 50
- haconf -makerw command 50
- had
 - definition 3
 - running on multiple systems 2
- hagrp -add command 68
- hagrp -modify command 68
- hagrp -offline command 59
- hagrp -resources command 54
- hagrp -unfreeze command 59
- halink daemon 3
- Hardware
 - communication channels 14
 - heartbeat disks 15
 - shared storage 13
- hares -clear command 62
- hares -dep command 53, 54
- hares -display command 53, 54
- hares -list command 52, 54, 55
- hares local command 63
- hares -modify command 72
- hares -offline command 61
- hares -offprop command 61
- hares -online command 61
- hares -probe command 62
- hastart command 48
- hastart -stale -force command 48
- hastatus command 57
- hastatus -group command 57
- hastop -all command 49
- hastop -local command 49
- hastop -sys command 49
- hatype -add command 66
- hatype -delete command 66
- hatype -modify command 66
- hauser -delete command 52
- hauser -modify command 51
- Heartbeat disks
 - configuring 15
 - definition 15
 - GAB 15

I

- IntentOnline attribute 207
- IPAddr attribute 224
- Istate attribute 199

L

- L1-A command 183
- LastOnline attribute 198
- LinkHbDown attribute 213
- LinkMonitoring attribute 220
- ListApplication attribute 202
- Load attribute 213
- LoadRaw attribute 213
- LogLevel attribute 217
- LogSize attribute 220

M

- main.cf 17
- MajorVersion attribute 213, 220
- ManualOps attribute 208
- MaxFactor attribute 220
- Membership services 15
- MinorVersion attribute 213, 221
- modify option 63
- MonitorInterval attribute 217
- MonitorOnly attribute 198
- MonitorTimeout attribute 217
- MountOpt attribute 201
- MountPoint attribute 201

N

- Name rule for resources 21
- NameRule attribute 218
- Network partitions
 - definition 182
 - how to prevent 182
 - preexisting 183
- NodeId attribute 213
- NumThreads attribute 218

O

- Object definitions
 - resources 26
 - service groups 26
 - SNMP 25
 - system 23
- OfflineTimeout attribute 218
- OnGrpCnt attribute 213
- OnlineRetryInterval attribute 208
- OnlineRetryLimit attribute 208, 218
- OnlineTimeout attribute 218



-
- OnlineWaitLimit attribute 218
 - OpenTimeout attribute 218
 - Operations attribute 219
- P**
- Parallel attribute 208
 - Parallel group 6
 - PartitionNo attribute 202
 - Port attribute 224
 - PreOnline attribute 209
 - PrintMsg attribute 221
 - PrintTree attribute 209
 - Probed attribute 199
 - ProbesPending attribute 209
 - ProcessClass attribute 222
 - ProcessPriority attribute 222
- R**
- ReadOnly attribute 221
 - Resource attributes 52, 54
 - Resource types
 - definitions 20
 - querying 55
 - ResourceLimit attribute 221
 - Resources
 - adding 72
 - attributes 27, 52, 54
 - bringing online 61
 - controlling 7
 - definition 6
 - deleting 74
 - dependencies 27
 - linking 73
 - localizing 63
 - monitoring 7
 - name rule 21
 - querying 52, 54
 - taking offline 61
 - RestartLimit attribute 219
- S**
- ScriptClass attribute 203, 219
 - ScriptPriority attribute 203, 219
 - Seeding
 - automatically 184
 - definition 184
 - manually 184
 - Service groups
 - adding 68
 - attributes 26
 - bringing online 59
 - configuring dependencies 144
 - deleting 74
 - disabling 60
 - disabling resources 60
 - enabling 60
 - enabling resources 60
 - failover group 6
 - flushing 61
 - freezing 59
 - heartbeat disk 15
 - modifying system lists 70
 - parallel group 6
 - switching 59
 - taking offline 59
 - thawing 59
 - Shared storage 13
 - SNMP predefined attributes 224
 - SourceFile attribute 209, 214, 219, 221
 - State attribute 200, 210
 - SysState attribute 215
 - System states 193
 - SystemList attribute 210
 - Systems
 - changing load attribute 58
 - displaying nodeid value 58
 - forcing to start 58
 - freezing 58
 - modifying attributes 58
 - querying 52
 - states 193
 - thawing 59
 - SystemZones attribute 210
- T**
- TFrozen attribute 210, 215
 - ToleranceLimit attribute 219
 - TrapList attribute 224
 - Type attribute 201
 - TypeDependencies attribute 211
 - TypeLimit attribute 221
 - types.cf 17
- U**
- UserInt attribute 215
 - UserIntGlobal attribute 211
 - UserIntLocal attribute 211
 - UserNames attribute 221



UserStr attribute 215
UserStrGlobal attribute 211
UserStrLocal attribute 211

V

VCS

- adding users 79
- core processes 3
- environment variables 45
- querying 52
- specifying priorities, scheduling classes 165
- starting on a system 47, 48
- stopping on a system 49

VCS configuration files

- components 18
- designating as stale 50
- how they define a cluster 18
- include clause 20
- main.cf 17
- read/write to read/only 50, 63

- removing stale designation 51
- types.cf 17

VCS configuration language

- definition 17
- high-level 75

VCS core processes

- agents 3
- GUI 3
- hacf 3
- halink 3

VCS server engine 3

- VCS_CONF environment variable 45
- VCS_GAB_PORT environment variable 45
- VCS_GAB_TIMEOUT environment variable 45
- VCS_HAD_RESTART_TIMEOUT environment variable 46
- VCS_HOME environment variable 45
- VCS_LOG environment variable 46
- VCS_PORT environment variable 46
- VCS_TEMP_DIR environment variable 46

