

Veritas™ Cluster Server Bundled Agents Reference Guide

Solaris

5.0 Maintenance Pack 3



Veritas Cluster Server Bundled Agents Reference Guide

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Product version: 5.0 MP3

Document version: 5.0MP3.0

Legal Notice

Copyright © 2008 Symantec Corporation. All rights reserved.

Symantec, the Symantec Logo, Veritas and Veritas Storage Foundation are trademarks or registered trademarks of Symantec Corporation or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation/reverse engineering. No part of this document may be reproduced in any form by any means without prior written authorization of Symantec Corporation and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. SYMANTEC CORPORATION SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

The Licensed Software and Documentation are deemed to be commercial computer software as defined in FAR 12.212 and subject to restricted rights as defined in FAR Section 52.227-19 "Commercial Computer Software - Restricted Rights" and DFARS 227.7202, "Rights in Commercial Computer Software or Commercial Computer Software Documentation", as applicable, and any successor regulations. Any use, modification, reproduction release, performance, display or disclosure of the Licensed Software and Documentation by the U.S. Government shall be solely in accordance with the terms of this Agreement.

Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014
<http://www.symantec.com>

Technical Support

Symantec Technical Support maintains support centers globally. Technical Support's primary role is to respond to specific queries about product features and functionality. The Technical Support group also creates content for our online Knowledge Base. The Technical Support group works collaboratively with the other functional areas within Symantec to answer your questions in a timely fashion. For example, the Technical Support group works with Product Engineering and Symantec Security Response to provide alerting services and virus definition updates.

Symantec's maintenance offerings include the following:

- A range of support options that give you the flexibility to select the right amount of service for any size organization
- Telephone and Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers automatic software upgrade protection
- Global support that is available 24 hours a day, 7 days a week
- Advanced features, including Account Management Services

For information about Symantec's Maintenance Programs, you can visit our Web site at the following URL:

www.symantec.com/techsupp

Contacting Technical Support

Customers with a current maintenance agreement may access Technical Support information at the following URL:

http://www.symantec.com/business/support/assistance_care.jsp

Before contacting Technical Support, make sure you have satisfied the system requirements that are listed in your product documentation. Also, you should be at the computer on which the problem occurred, in case it is necessary to replicate the problem.

When you contact Technical Support, please have the following information available:

- Product release level
- Hardware information
- Available memory, disk space, and NIC information
- Operating system
- Version and patch level
- Network topology
- Router, gateway, and IP address information
- Problem description:

- Error messages and log files
- Troubleshooting that was performed before contacting Symantec
- Recent software configuration changes and network changes

Licensing and registration

If your Symantec product requires registration or a license key, access our technical support Web page at the following URL:

www.symantec.com/techsupp

Customer service

Customer service information is available at the following URL:

www.symantec.com/techsupp

Customer Service is available to assist with the following types of issues:

- Questions regarding product licensing or serialization
- Product registration updates, such as address or name changes
- General product information (features, language availability, local dealers)
- Latest information about product updates and upgrades
- Information about upgrade assurance and maintenance contracts
- Information about the Symantec Buying Programs
- Advice about Symantec's technical support options
- Nontechnical presales questions
- Issues that are related to CD-ROMs or manuals

Documentation feedback

Your feedback on product documentation is important to us. Send suggestions for improvements and reports on errors or omissions to

clustering_docs@symantec.com.

Include the title and document version (located on the second page), and chapter and section titles of the text on which you are reporting.

Maintenance agreement resources

If you want to contact Symantec regarding an existing maintenance agreement, please contact the maintenance agreement administration team for your region as follows:

Asia-Pacific and Japan	contractsadmin@symantec.com
Europe, Middle-East, and Africa	semea@symantec.com
North America and Latin America	supportsolutions@symantec.com

Additional enterprise services

Symantec offers a comprehensive set of services that allow you to maximize your investment in Symantec products and to develop your knowledge, expertise, and global insight, which enable you to manage your business risks proactively.

Enterprise services that are available include the following:

Symantec Early Warning Solutions	These solutions provide early warning of cyber attacks, comprehensive threat analysis, and countermeasures to prevent attacks before they occur.
Managed Security Services	These services remove the burden of managing and monitoring security devices and events, ensuring rapid response to real threats.
Consulting Services	Symantec Consulting Services provide on-site technical expertise from Symantec and its trusted partners. Symantec Consulting Services offer a variety of prepackaged and customizable options that include assessment, design, implementation, monitoring, and management capabilities. Each is focused on establishing and maintaining the integrity and availability of your IT resources.
Educational Services	Educational Services provide a full array of technical training, security education, security certification, and awareness communication programs.

To access more information about Enterprise services, please visit our Web site at the following URL:

www.symantec.com

Select your country or language from the site index.

Contents

Chapter 1	Introduction	
	Resources and their attributes	19
	Modifying agents and their resources	20
	Attributes	20
Chapter 2	Storage agents	
	About the storage agents	23
	DiskGroup agent	24
	Dependencies	24
	Agent functions	24
	State definitions	26
	Attributes	26
	Resource type definition	28
	DiskGroup agent notes	29
	High availability fire drill	29
	Using volume sets in Solaris	29
	Setting the noautoimport flag for a disk group	29
	Configuring the Fiber Channel adapter	30
	Sample configurations	30
	DiskGroup resource configuration	30
	DiskGroupSnap agent	31
	Dependencies	31
	Agent functions	32
	State definitions	32
	Attributes	32
	DiskGroupSnap agent notes	34
	Configuration considerations	34
	Agent limitations	34
	Resource type definition	35
	Sample configurations	35
	Volume agent	38
	Dependencies	38
	Agent functions	38
	State definitions	39
	Attributes	39

Resource type definition	39
Sample configurations	39
Configuration	39
Mount agent	40
Dependencies	40
Agent functions	40
State definitions	42
Attributes	43
Resource type definition	46
Mount agent notes	47
High availability fire drill	47
VxFS file system lock	47
ZFS file system and pool creation example	47
Sample configurations	48
VxFS configuration example	48
Zpool agent	49
Limitations	49
Dependencies	50
Agent functions	50
State definitions	51
Attributes	51
Resource type definition	53
Sample configurations	53

Chapter 3 Network agents

About the network agents	55
Agent comparisons	56
IP and NIC agents	56
IPMultiNIC and MultiNICA agents	56
IPMultiNICB and MultiNICB agents	56
802.1Q trunking	57
IP agent	58
High availability fire drill	58
Dependencies	58
Agent functions	59
State definitions	59
Attributes	60
Resource type definition	62
Sample configurations	63
Configuration 1	63
NetMask in decimal (base 10)	63
Configuration of NetMask in hexadecimal (base 16)	63

NIC agent	64
High availability fire drill	64
Dependencies	64
Agent functions	65
State definitions	65
Attributes	66
Resource type definition	67
Sample configurations	67
Configuration without network hosts (using default ping mechanism)	67
Configuration with network hosts	67
IPMultiNIC agent	68
Dependencies	68
Agent functions	68
State definitions	69
Attributes	69
Resource type definition	71
Sample configuration: IPMultiNIC and MultiNICA	71
MultiNICA agent	72
Dependencies	72
Agent function	72
State definitions	73
Attributes	73
Resource type definition	76
MultiNICA notes	76
Using RouteOptions	77
Sample configurations	77
MultiNICA and IPMultiNIC	77
About the IPMultiNICB and MultiNICB agents	79
Checklist to ensure the proper operation of MultiNICB	79
IPMultiNICB agent	80
Dependencies	80
Requirements for IPMultiNICB	80
Agent functions	81
State definitions	81
Attributes	82
Resource type definition	84
Manually migrating a logical IP address Solaris	84
Sample configurations	84
Other sample configurations for IPMultiNICB and MultiNICB	84
MultiNICB agent	85
Base and Multipathing modes	85
Sun trunking	85

Dependencies	85
Agent functions	86
State definitions	86
Attributes	87
Optional attributes for Base and Mpathd modes	87
Optional attributes for Base mode	89
Optional attributes for Multipathing mode	92
Resource type definition	93
Solaris operating modes: Base and Multipathing	93
Base mode	93
Multipathing mode	95
Trigger script	95
Sample configurations	96
Interface configuration	96
Setting up test IP addresses for Base Mode	96
IPMultiNICB and MultiNICB configuration	97
DNS agent	98
Dependencies	98
Agent functions	99
State definitions	100
Attributes	101
Resource type definition	105
DNS agent notes	105
High availability fire drill	105
Monitor scenarios	106
Sample Web server configuration	106
Secure DNS update for BIND 9	106
Setting up secure updates using TSIG keys for BIND 9	107

Chapter 4 File share agents

About the file service agents	109
NFS agent	110
Dependencies	110
Agent functions	111
State definitions	111
Attributes	112
Resource type definition	112
NFS agent notes	113
Service Management Facility for Solaris 10	113
Sample configurations	114
Configuration	114

NFSRestart agent	115
Dependencies	115
Agent functions	116
State definitions	116
Attributes	117
Resource type definition	117
NFSRestart agent notes	118
High availability fire drill	118
Service Management Facility–Solaris 10	118
Providing a fully qualified host name	120
Sample configurations	121
Share agent	123
Dependencies	123
Agent functions	123
State definitions	124
Attributes	124
Resource type definition	124
Share agent notes	125
High availability fire drill	125
Sample configurations	125
Configuration	125
About the Samba agents	126
The Samba agents	126
Before using the Samba agents	126
Supported versions	127
Configuring the Samba agents	127
SambaServer agent	128
Dependencies	128
Agent functions	128
State definitions	129
Attributes	129
Resource type definitions	130
Sample configurations	130
SambaShare agent	131
Dependencies	131
Agent functions	131
State definitions	132
Attributes	132
Resource type definition	133
Sample configuration	133
NetBIOS agent	134
Dependencies	134
Agent functions	135

State definitions	135
Attributes	136
Resource type definition	137
Sample configuration	137

Chapter 5 Service and application agents

About the service and application agents	139
Apache Web server agent	140
Dependencies	140
Agent functions	141
State definitions	141
Attributes	142
Resource type definition	146
Apache Web server notes	147
Tasks to perform before you use the Apache Web server agent	147
Detecting application failure	148
About bringing an Apache Web server online outside of VCS control	148
About the ACC Library	149
High Availability fire drill	149
Sample configurations	149
Application agent	151
High availability fire drill	151
Dependencies	151
Agent functions	152
State definitions	153
Attributes	154
Resource type definition	158
Sample configurations	158
Configuration 1	158
Configuration 2	158
Configuration 3 for Solaris 10	159
Process agent	160
High availability fire drill	160
Dependencies	160
Agent functions	161
State definitions	161
Attributes	162
Resource type definition	163
Sample configurations	163
Configuration 1	163
Configuration 2	163

ProcessOnOnly agent	164
Dependencies	164
Agent functions	164
State definitions	164
Attributes	165
Resource type definition	166
Sample configurations	166
Zone agent	167
Dependencies	167
Agent functions	167
Attributes	168
Resource type definition	168
Sample configuration	169
Configuration for Solaris 10	169
LDom agent	170
Limitations	170
Dependencies	170
Network resources	170
Storage resources	170
Agent functions	171
State definitions	171
Attributes	172
Resource type definition	173
Sample configuration	173

Chapter 6 Infrastructure and support agents

About the infrastructure and support agents	175
NotifierMngr agent	176
Dependency	176
Agent functions	176
State definitions	176
Attributes	177
Resource type definition	180
Sample configuration	181
Configuration	181
VRTSWebApp agent	183
Agent functions	183
State definitions	183
Attributes	184
Resource type definition	184
Sample configuration	185

Proxy agent	186
Dependencies	186
Agent functions	186
Attributes	187
Resource type definition	188
Sample configurations	188
Configuration 1	188
Configuration 2	188
Configuration 3	188
Phantom agent	190
Dependencies	190
Agent functions	190
Attribute	190
Resource type definition	190
Sample configurations	191
Configuration 1	191
Configuration 2	191
RemoteGroup agent	192
Dependency	192
Agent functions	193
State definitions	193
Attributes	194
Resource type definition	198

Chapter 7

Testing agents

About the program support agents	199
ElifNone agent	200
Dependencies	200
Agent function	200
Attributes	201
Resource type definition	201
Sample configuration	201
FileNone agent	202
Dependencies	202
Agent functions	202
Attribute	203
Resource type definition	203
Sample configuration	203

FileOnOff agent	204
Dependencies	204
Agent functions	204
Attribute	205
Resource type definition	205
Sample configuration	205
FileOnOnly agent	206
Dependencies	206
Agent functions	206
Attribute	207
Resource type definition	207
Sample configuration	207
Glossary	209
Index	211

Introduction

Bundled agents are Veritas Cluster Server (VCS) processes that manage resources of predefined resource types according to commands received from the VCS engine, HAD. You install these agents when you install VCS.

A node has one agent per resource type that monitors all resources of that type. For example, a single IP agent manages all IP resources.

When the agent starts, it obtains the necessary configuration information from VCS. The agent then periodically monitors the resources, and updates VCS with the resource status.

Agents can:

- Bring resources online.
- Take resources offline.
- Monitor resources and report state changes.

For a more detailed overview of agents, see the VCS User's Guide.

Resources and their attributes

Resources are parts of a system and are known by their type, such as: a volume, a disk group, or an IP address. VCS includes a set of resource types. Different attributes define these resource types in the `types.cf` file. Each type has a corresponding agent that controls the resource.

The VCS configuration file, `main.cf`, contains the values for the resource attributes and has an `include` directive to the `types.cf` file.

An attribute's given value configures the resource to function in a specific way. By modifying the value of a resource attribute, you can change the way the VCS agent manages the resource. For example, the IP agent uses the `Address` attribute to determine the IP address to monitor.

Modifying agents and their resources

Use the Cluster Manager (Java Console), Veritas Cluster Server Management Console, or the command line to dynamically modify the configuration of the resources managed by an agent.

See the *Veritas Cluster Server User's Guide* for instructions on how to complete these tasks.

VCS enables you to edit the main.cf file directly. To implement these changes, make sure to restart VCS.

Attributes

Attributes contain data about the cluster, systems, service groups, resources, resource types, and the agent. An attribute has a definition and a value. You change attribute values to configure VCS resources. Attributes are either optional or required, although sometimes attributes that are optional in one configuration might be required in other configurations. Many optional attributes have predefined or default values, which you should change as required.

A variety of internal use only attributes also exist. Do not modify these attributes—modifying them can lead to significant problems for your clusters.

Attributes have type and dimension. Some attribute values can accept numbers, others can accept alphanumeric values or groups of alphanumeric values, while others are simple boolean on/off values.

Table 1-1 Attribute data types

Data Type	Description
string	Enclose strings, which are a sequence of characters, in double quotes ("). Optionally enclose strings in quotes when they begin with a letter, and contains only letters, numbers, dashes (-), and underscores (_). A string can contain double quotes, but the quotes must be immediately preceded by a backslash. In a string, represent a backslash with two slashes (//).
integer	Signed integer constants are a sequence of digits from 0 to 9. You can precede them with a dash. They are base 10. Integers cannot exceed the value of a 32-bit signed integer: 21471183247.

Table 1-1 Attribute data types

Data Type	Description
boolean	A boolean is an integer with the possible values of 0 (false) and 1 (true).

Table 1-2 Attribute dimensions

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. Find the specified brackets after the attribute name on the attribute definition in the types.cf file.
keylist	A keylist is an unordered list of unique strings.
association	An association is an unordered list of name-value pairs. An equal sign separates each pair. A set of braces ({}) denotes that an attribute is an association. Braces are specified after the attribute name on the attribute definition in the types.cf file, for example: str SnmpConsoles{}.

Storage agents

This chapter contains:

- [“DiskGroup agent”](#) on page 24
- [“DiskGroupSnap agent”](#) on page 31
- [“Volume agent”](#) on page 38
- [“Mount agent”](#) on page 40
- [“Zpool agent”](#) on page 49

About the storage agents

Use storage agents to Monitor shared storage.

DiskGroup agent

The DiskGroup agent brings online, takes offline, and monitors Veritas Volume Manager (VxVM) disk groups. This agent uses VxVM commands. You can use this agent to monitor or make disk groups highly available.

When the value of the StartVolumes and StopVolumes attribute is 1, the DiskGroup agent brings the volumes online and takes them offline during the import and deport operations of the disk group.

When you use a volume set, set StartVolumes and StopVolumes attributes of the DiskGroup resource that contains the volume set to 1. If a file system is created on the volume set, use a Mount resource to mount the volume set.

The agent protects data integrity by disabling failover when data is written to a volume in the disk group.

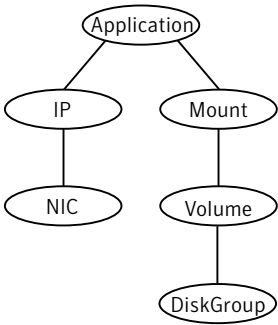
For important information on this agent, refer to:

“[DiskGroup agent notes](#)” on page 29

Dependencies

The DiskGroup resource does not necessarily depend on any other resource.

Figure 2-1 Sample service group for a DiskGroup resource



Agent functions

Online	Imports the disk group using the vxdg command.
Offline	Deports the disk group using the vxdg command.

Monitor	Determines if the disk group is online or offline using the <code>vxdbg</code> command. <pre># vxdbg -g disk_group set autoimport=no</pre>
Clean	Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.
Info	The DiskGroup info agent function gets information from the Volume Manager and displays the type and free size for the DiskGroup resource. Initiate the info agent function by setting the InfoInterval timing to a value greater than 0. In the following example, the info agent function executes every 60 seconds: <pre># haconf -makerw # hatype -modify DiskGroup InfoInterval 60</pre> The command to retrieve information about the DiskType and FreeSize of the DiskGroup resource is: <pre># hares -value diskgroupres ResourceInfo</pre> Output includes: <pre>DiskType sliced FreeSize 35354136</pre>
Action	Different action agent functions follow: ■ <code>license.vfd</code> Checks for valid Veritas Volume manager license—if one is not found use the <code>vxlicinst</code> utility to install a valid license key. ■ <code>disk.vfd</code> Checks if all disks in diskgroup are visible on host—if it fails, check if the path to disks exists from the host and check if LUN masking and zoning are set properly. ■ <code>udid.vfd</code> Checks the UDIDs of disks on the cluster nodes—if it fails, ensure that the disks that are used for the disk group are the same on all cluster nodes. ■ <code>verifyplex.vfd</code> Checks if the number of plexes on each site for the Campus Cluster setup are set properly—if it fails, check that the sites, disks, and plexes are set properly for a Campus Cluster setup. ■ <code>volinuse</code> Checks if open volumes are in use or file systems on volumes that are mounted outside of VCS configuration. See “High availability fire drill” on page 29.

State definitions

ONLINE	Indicates that the disk group is imported.
OFFLINE	Indicates that the disk group is not imported.
FAULTED	Indicates that the disk group has unexpectedly deported or become disabled.
UNKNOWN	Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

Attributes

Table 2-1 Required attributes

Required attribute	Description
DiskGroup	Name of the disk group that is configured with Veritas Volume Manager. Type and dimension: string-scalar

Table 2-2 Optional attributes

Optional attributes	Description
MonitorReservation	If the value is 1, and SCSI-3 fencing is used, the agent monitors the SCSI reservation on the disk group. If the reservation is missing, the monitor agent function takes the resource offline. Type and dimension: boolean-scalar Default: 0

Table 2-2 Optional attributes

Optional attributes	Description
PanicSystemOnDGLoss	Determines whether to panic the node if the disk group becomes disabled. A loss of storage connectivity can cause the disk group to become disabled. If the value of this attribute is 1 and the disk group becomes disabled, the node panics. If the value of the attribute is 0 and the disk group becomes disabled, the following occurs: ■ If the cluster has I/O fencing enabled, the DiskGroup resource is marked <code>FAULTED</code>. This state results in the agent attempting to take the service group offline. As part of bringing the DiskGroup resource offline, the agent attempts to deport the disabled disk group. Even if disabled disk group fails to deport, the DiskGroup resource enters a <code>FAULTED</code> state. This state enables the failover of the service group that contains the resource. To fail back the DiskGroup resource, manually deport the disk group after restoring storage connectivity ■ If the cluster does not use I/O fencing, a message is logged and the resource is reported <code>ONLINE</code>. Type and dimension: boolean-scalar Default: 1
StartVolumes	If value is 1, the DiskGroup online function starts all volumes belonging to that disk group after importing the group. Type and dimension: string-scalar Default: 1
StopVolumes	If value is 1, the DiskGroup offline function stops all volumes belonging to that disk group before it deports the group. Type and dimension: string-scalar Default: 1

Table 2-2 Optional attributes

Optional attributes	Description
UmountVolumes	This attribute enables the DiskGroup resource to forcefully go offline even if open volumes are mounted outside of VCS control. When the value of this attribute is 1 and the disk group has open volumes, the following occurs: ■ The agent attempts to unmount the file systems on open volumes. If required, the agent attempts to kill all VCS managed and un-managed applications using the file systems on those open volumes. ■ The agent attempts to forcefully unmount the file systems to close the volumes. Type and dimension: integer-scalar Default: 0
TempUseFence	Do not use. For internal use only.
DiskGroupType	Do not use. For internal use only.

Resource type definition

```
type DiskGroup (  
    static keylist SupportedActions = { "license.vfd", "disk.vfd",  
    "udid.vfd", "verifyplex.vfd", checkudid, numdisks, campusplex,  
    joindg, splitdg, getvxvminfo, volinuse }  
    static int OnlineRetryLimit = 1  
    static str ArgList[] = { DiskGroup, StartVolumes, StopVolumes,  
    UmountVolumes, MonitorOnly, MonitorReservation, tempUseFence,  
    PanicSystemOnDGLoss }  
    str DiskGroup  
    str StartVolumes = 1  
    str StopVolumes = 1  
    int UmountVolumes = 0  
    static int NumThreads = 1  
    boolean MonitorReservation = 0  
    temp str tempUseFence = INVALID  
    boolean PanicSystemOnDGLoss = 1  
)
```

DiskGroup agent notes

The DiskGroup agent has the following notes:

- [“High availability fire drill”](#) on page 29
- [“Using volume sets in Solaris”](#) on page 29
- [“Setting the noautoimport flag for a disk group”](#) on page 29
- [“Configuring the Fiber Channel adapter”](#) on page 30

High availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node. These discrepancies might prevent a service group from going online on a specific node.

For DiskGroup resources, the high availability fire drill checks for:

- The Veritas Volume Manager license
- Visibility from host for all disks in the disk group
- The same disks for the disk group on cluster nodes
- Equal number of plexes on all sites for the disk group in a campus cluster setup

For more information about using the high availability fire drill see the *Veritas Cluster Server User's Guide*.

Using volume sets in Solaris

When you use a volume set, set StartVolumes and StopVolumes attributes of the DiskGroup resource that contains a volume set to 1. If a file system is created on the volume set, use a Mount resource to mount the volume set.

See the UNIX Mount agent for more information.

Setting the noautoimport flag for a disk group

VCS requires that the noautoimport flag of an imported disk group be explicitly set to true. This value enables VCS to control the importation and deportation of disk groups as needed when bringing disk groups online and taking them offline.

To check the status of the noautoimport flag for an imported disk group

◆ `# vxprint -l disk_group | grep noautoimport`

If the output from this command is blank, the noautoimport flag is set to false and VCS lacks the necessary control.

VxVM versions 4.1 and 5.0

The Monitor function changes the value of the VxVM noautoimport flag from off to on. It changes the value instead of taking the service group offline. This action allows VCS to maintain control of importing the disk group.

The following command changes the autoimport flag to false:

```
# vxdbg -g disk_group set autoimport=no
```

For VxVM version 4.0

When you enable a disk group that is configured as a DiskGroup resource that does not have the noautoimport flag set to true, VCS forcibly deports the disk group. This forcible deportation may disrupt applications running on the disk group.

To explicitly set the noautoimport flag to true, deport the disk group and import it with the -t option as follows:

To deport the disk group, enter:

```
# vxdbg deport disk_group
```

To import the disk group, specifying the noautoimport flag be set to true to ensure that the disk group is not automatically imported, enter:

```
# vxdbg -t import disk_group
```

Configuring the Fiber Channel adapter

Most Fiber Channel (FC) drivers have a configurable parameter called “failover.” This configurable parameter is in the FC driver’s configuration file. This parameter is the number of seconds that the driver waits before it transitions a disk target from OFFLINE to FAILED. After the state becomes FAILED, the driver flushes all pending fiber channel commands back to the application with an error code. Symantec recommends that you use a non-zero value that is smaller than any of the MonitorTimeout values of the Disk Group resources. Use this value to avoid excessive waits for monitor timeouts.

Refer to the Fiber Channel adapter's configuration guide for further information.

Sample configurations

DiskGroup resource configuration

Example of a disk group resource in the Share Out mode.

```
DiskGroup dgl (
    DiskGroup = testdg_1
)
```

DiskGroupSnap agent

Use the DiskGroupSnap agent to perform fire drills in a campus cluster. The DiskGroupSnap agent enables you to verify the configuration and data integrity in a Campus Cluster environment (with VxVM stretch mirroring).

For more information on fire drills, refer to the *Veritas Cluster Server User's Guide*.

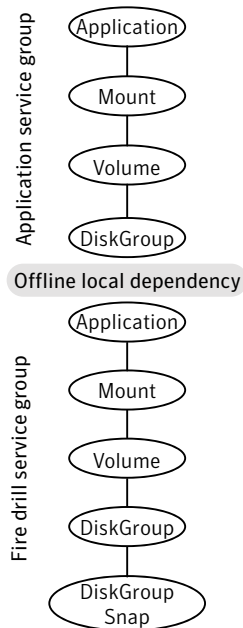
For important information about this agent, refer to:

“[DiskGroupSnap agent notes](#)” on page 34

Dependencies

The DiskGroupSnap resource does not depend on any other resources. The service group that contains the DiskGroupSnap agent has an offline local dependency on the application's service group.

Figure 2-2 Sample service group for a DiskGroupSnap resource



Agent functions

Online	Verifies that the application's disk group is in a valid campus cluster configuration. It detaches the site that the value of the FDSiteName attribute specifies. It then creates another disk group to be used for the fire drill on the detached site.
Offline	This re-attaches the site that the value of the FDSiteName attribute specifies back to the application's disk group.
Monitor	Monitors the DiskGroupSnap resource.
Clean	Takes the DiskGroupSnap resource offline.
Open	If the DiskGroupSnap resource has a parent resource that is not ONLINE, then it deletes the online lock file of the DiskGroupSnap resource. This marks the DiskGroupSnap resource as OFFLINE. In all other cases, the DiskGroupSnap resource performs no action.

State definitions

ONLINE	The DiskGroupSnap resource functions normally.
OFFLINE	The DiskGroupSnap resource is not running.
UNKNOWN	A configuration error exists.

Attributes

Table 2-3 Required attributes

Required attribute	Description
TargetResName	The name of the DiskGroup resource from the application's service group. Type-dimension: string-scalar Example: "dgres1"

Table 2-3 Required attributes

Required attribute	Description
FDSiteName	This is the site name that fire drill disks use. This name must be distinct for each site. You need to assign this local (per system) values as it maps to the SystemZone of the application service group. For more information about the SystemZone attribute, refer to the <i>Veritas Cluster Server User's Guide</i>. You can run the fire drill in the following two configurations: ■ Use a dedicated set of disks at the secondary that have been set aside for fire drill use. In this case, you must set the FDSiteName attribute to the VxVM site name given to this set of disks. This setup is commonly referred to as the Gold configuration. ■ Use the same disks that make up the mirror at the secondary site. In this case, you must set the FDSiteName attribute to the VxVM site name of the secondary site. This setup is commonly referred to as the Bronze configuration. Type and dimension: string-scalar Example: When the application service group has the following values for the SystemZones attribute: SystemZones = { n1 = 0, n2 = 0, n3 = 1, n4 = 1 } Where n1 (node 1) and n2 (node 2) comprise the first site and where the second site has n3 (node 3) and n4 (node 4). The FDSiteName definitions in the fire drill service group resemble the following: ■ "FDSiteName@n1=fdpri" ■ "FDSiteName@n2=fdpri" ■ "FDSiteName@n3=fdsec" ■ "FDSiteName@n4=fdsec" The fdpri and fdsec values are the site names of dedicated fire drill site disks at the primary and secondary sites respectively.

DiskGroupSnap agent notes

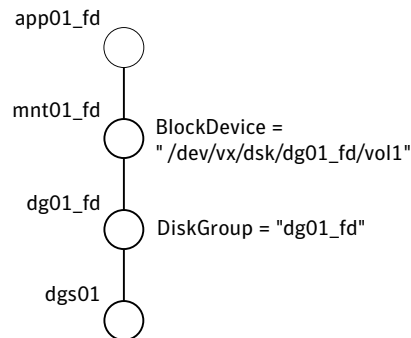
Configuration considerations

Keep the following recommendations in mind:

- Do not bring the DiskGroupSnap resource online in the SystemZone where the application service group is online.
- When you create the fire drill service group, in general use the same attribute values that you use in the application service group. However, the BlockDevice attribute of the Mount resource and the DiskGroup attribute of the DiskGroup resource change between the application's service group and the fire drill's service group. You must append an `_fd` to the original disk group name for the disk group name that the fire drill uses. For example, if `dg01` is the disk group's name in the application service group, the attributes in the fire drill resemble those in [Figure 2-3](#).

[Figure 2-3](#) shows the changes to resource values for the fire drill service group; note that the Volume resource is not included.

Figure 2-3 Sample resource values for a DiskGroupSnap resource



Agent limitations

The following limitations apply to the DiskGroupSnap agent:

- The online and offline operations of the DiskGroupSnap resource invokes VCS action entry points to run VxVM commands to detach/reattach the fire drill site. Since VxVM requires that these commands are run on the node where the disk group is imported, the disk group has to be imported on some node in the cluster before these operations.

- If you attempt to shut down the node where you brought the fire drill service group online, the node goes to a LEAVING state and the VCS engine attempts to take all the service groups offline on that node. At this point, the VCS engine rejects all action entry point requests. Therefore, during offline the DiskGroupSnap agent cannot invoke the action to reattach the fire drill site to the target diskgroup. The agent logs a message that the node is in a leaving state and then removes the lock file. The agent's monitor function declares that the resource is offline. After the node restarts, the fire drill site still remains detached from the diskgroup and you must manually reattach it.
- If you halt the node while the DiskGroupSnap resource's service group is still online, the VxVM site used for the fire drill remains detached after the node is brought up. You must manually reattach the fire drill site to the original diskgroup at the primary site.
- Before you shut down or stop VCS locally on the node where the fire drill service group is online, you must take the fire drill service group offline. Otherwise, after the node restarts you must manually reattach the fire drill site to the disk group that is imported at the primary site.

Resource type definition

```
type DiskGroupSnap (
    static int ActionTimeout = 120
    static int MonitorInterval = 300
    static int NumThreads = 1
    static str ArgList[] = { TargetResName, FDSiteName }
    str TargetResName
    str FDSiteName
)
```

Sample configurations

The following sample configure shows the fire drill's service group and its corresponding application service group. The fire drill's service group follows:

```
group dgfdsg (
    SystemList = { thoribm32 = 0, thoribm31 = 1 }
    SystemZones = { thoribm32 = 1, thoribm31 = 0 }
)

DiskGroup dgfdres (
    DiskGroup = newdgl_fd
)

DiskGroupSnap dgsres (
    TargetResName = dgres
```

```
FDSiteName @thoribm32 = firedrill
FDSiteName @thoribm31 = firedrill_31
)

Mount mntfdres1 (
  MountPoint = "/dgsfs1"
  BlockDevice = "/dev/vx/dsk/newdg1_fd/newvol1"
  FSType = vxfs
  FsckOpt = "-y"
)

Mount mntfdres2 (
  MountPoint = "/dgsfs2"
  BlockDevice = "/dev/vx/dsk/newdg1_fd/newvol2"
  FSType = vxfs
  FsckOpt = "-y"
)

Process procfres1 (
  PathName = "/usr/bin/ksh"
  Arguments = "/scrib.sh /dgsfs1"
)

Process procfres2 (
  PathName = "/usr/bin/ksh"
  Arguments = "/scrib.sh /dgsfs2"
)

requires group dgsg offline local
dgfdres requires dgsres
mntfdres1 requires dgfdres
mntfdres2 requires dgfdres
procfres1 requires mntfdres1
procfres2 requires mntfdres2
```

The application's service group follows:

```
group dgsg (
  SystemList = { thoribm32 = 0, thoribm31 = 1 }
  SystemZones = { thoribm31 = 0, thoribm32 = 1 }
)

DiskGroup dgres (
  DiskGroup = newdg1
)

Mount mntres1 (
  MountPoint = "/dgsfs1"
  BlockDevice = "/dev/vx/dsk/newdg1/newvol1"
  FSType = vxfs
  FsckOpt = "-y"
)
```

```
Mount mntres2 (  
  MountPoint = "/dgsfs2"  
  BlockDevice = "/dev/vx/dsk/newdgl/newvol2"  
  FSType = vxfs  
  FsckOpt = "-y"  
)  
  
Process procrs1 (  
  PathName = "/usr/bin/ksh"  
  Arguments = "/scrib.sh /dgsfs1"  
)  
  
Process procrs2 (  
  PathName = "/usr/bin/ksh"  
  Arguments = "/scrib.sh /dgsfs2"  
)  
  
mntres1 requires dgres  
mntres2 requires dgres  
procrs1 requires mntres1  
procrs2 requires mntres2
```

Volume agent

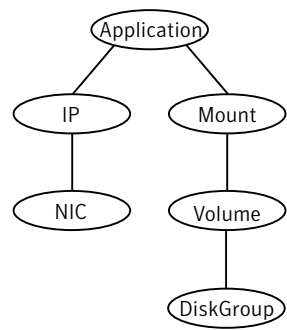
The Volume agent brings online, takes offline, and monitors a Veritas Volume Manager (VxVM) volume. You can use the agent to make a volume highly available or to monitor it.

Note: Do not use the Volume agent for volumes created for replication.

Dependencies

Volume resources depend on DiskGroup resources.

Figure 2-4 Sample service group for a Volume resource



Agent functions

Online	Starts the volume using the <code>vxrecover</code> command.
Offline	Stops the volume using the <code>vxvol</code> command.
Monitor	Determines if the volume is online or offline by reading a block from the raw device interface to the volume.
Clean	Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.

State definitions

ONLINE	Indicates that the specified volume is started and that I/O is permitted.
OFFLINE	Indicates that the specified volume is not started and that I/O is not permitted.
FAULTED	Indicates the volume stops unexpectedly.
UNKNOWN	Indicates that the agent could not determine the state of the resource or that the resource attributes are invalid.

Attributes

Table 2-4 Required attributes

Required attribute	Description
DiskGroup	Name of the disk group that contains the volume. Type and dimension: string-scalar
Volume	Name of the volume from disk group specified in DiskGroup attribute. Type and dimension: string-scalar

Resource type definition

```
type Volume (  
    static str ArgList[] = { Volume, DiskGroup }  
    str Volume  
    str DiskGroup  
    static int NumThreads = 1  
)
```

Sample configurations

Configuration

```
Volume sharedg_vol3 (  
    Volume = vol3  
    DiskGroup = sharedg  
)
```

Mount agent

The Mount agent brings online, takes offline, and monitors a file system or an NFS client mount point. You can use the agent to make file systems or NFS client mount points highly available or to monitor them. This agent also supports high availability fire drills.

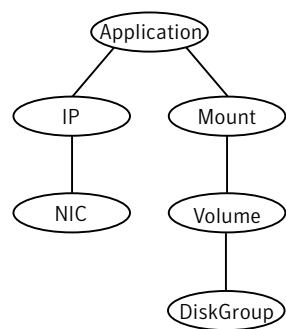
For important information about this agent, refer to:

“Mount agent notes” on page 47

Dependencies

No dependencies exist for the Mount resource.

Figure 2-5 Sample service group for a Mount resource



Agent functions

Online	Mounts a block device on the directory. If the mount process fails for non-NFS mounts, the agent attempts to run the <code>fsck</code> command on the device to remount the block device. If file system type is NFS, agent mounts the remote file system to a specified directory. The remote NFS file system is specified in the <code>BlockDevice</code> attribute.
Offline	Unmounts the mounted file system gracefully.
Monitor	Determines if the file system is mounted.
Clean	Unmounts the mounted file system forcefully.

Info

The Mount info agent function executes the command:

```
df -k mount_point
```

The output displays Mount resource information:

```
Size Used Avail Use%
```

To initiate the info agent function, set the InfoInterval timing to a value greater than 0. In this example, the info agent function executes every 60 seconds:

```
haconf -makerw
```

```
hatype -modify Mount InfoInterval 60
```

The command to retrieve information about the Mount resource is:

```
hares -value mountres ResourceInfo
```

Output includes:

```
Size 2097152  
Used 139484  
Available 1835332  
Used% 8%
```

Action

- **chgmtlock**
Invoke this action to reset the VxFS file system lock to a VCS-defined lock.
- **mountpoint.vfd**
Checks if the specified mount point exists on the offline node. If it fails, it creates the mount point directory using `mkdir` command.
- **mounted.vfd**
Checks if the mount point is already mounted on the offline node. If it fails, you need to unmount all the file systems from the specified mount point directory.
- **vxfslic.vfd**
Checks for valid Veritas File System (VxFS) licenses. If it fails, you need to update the license for VxFS.
- **mountentry.vfd**
Checks that the mount point is not listed in file system tables (e.g. `/etc/vfstab`).
This action prevents the automatic mounting of the file system when the system reboots. If it fails, you need to remove mount point from file system tables.

State definitions

ONLINE	For the local file system, indicates that the block device is mounted on the specified mount point. For an NFS client, indicates that the NFS remote client is mounted on the specified mount directory.
OFFLINE	For the local file system, indicates that the block device is not mounted on the specified mount point. For an NFS client, indicates that the NFS remote client is not mounted on the specified mount directory.
FAULTED	For the local file system, indicates that the block device has unexpectedly unmounted. For the NFS client, indicates that the NFS remote client has unexpectedly unmounted.
UNKNOWN	Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

Attributes

Table 2-5 Required attributes

Required attribute	Description
BlockDevice	Block device for mount point. Type and dimension: string-scalar Example: <code>"/dev/vx/dsk/myvcs_dg/myvol"</code>
FsckOpt	Mandatory for non-NFS mounts. Use this attribute to specify options for the <code>fsck</code> command. You must correctly set this attribute for local mounts. If the mount process fails, the <code>fsck</code> command is executed with the specified options before it attempts to remount the block device. Its value must include either <code>-y</code>, <code>-n</code>, or <code>-p</code>. Refer to the <code>fsck</code> manual page for more information. For NFS mounts, the value of this attribute is not applicable and is ignored. Type and dimension: string-scalar VxFS example: <code>-y</code>
FSType	Type of file system. Supports <code>ufs</code>, <code>nfs</code>, <code>zfs</code>, or <code>vxfs</code>. Type and dimension: string-scalar Example: <code>"vxfs"</code>
MountPoint	Directory for mount point Type and dimension: string-scalar Example: <code>"/tmp/mnt"</code>

Table 2-5 Required attributes

Required attribute	Description
VxFSMountLock	This attribute is only applicable to Veritas (VxFS) file systems. This attribute controls a file system locking feature to prevent accidental unmounts. This attribute can take three values: 0, 1, or 2. VxFSMountLock=0 The resource does not detect any changes to the lock when VCS reports that it is online after you set the value to zero. ■ If the mount point is initially locked with the mntlock="VCS", the monitor agent function unlocks it. ■ If the mount point is initially locked with a key that is not equal to "VCS", a message is logged once or the agent logs a message once. ■ If the mount point is initially not locked, no action is performed. VxFSMountLock=1 The resource does not detect changes to the lock when VCS reports it online after the value was set to one. VCS does not monitor the lock. ■ If the mount point is initially locked with the mntlock="VCS", no action is performed. ■ If the mount point is initially locked with a key that is not equal to "VCS", a message is logged once or the agent logs a message once. ■ If the mount point is initially not locked, the monitor agent function locks it with the mntlock="VCS". VxFSMountLock=2 When the value of the VxFSMountLock is 2, the file system is locked and the agent monitors any change to mntlock. ■ If the mount point is locked with the mntlock="VCS", no action is performed. ■ If the mount point is initially locked with a key that is not equal to "VCS", the monitor agent function logs a message whenever a change in mntlock is detected. ■ If the mount point is not locked, the agent locks it with the mntlock="VCS". Type and dimension: integer-scalar Default: 1

Table 2-6 Optional attributes

Optional attribute	Description
MountOpt	Options for the <code>mount</code> command. Refer to the <code>mount</code> manual page for more information. Do not set the VxFS mount option "<code>mntlock=key</code>". The agent uses this option only when bringing a Mount resource online. Type and dimension: string-scalar Example: "<code>rw</code>"
SnapUmount	If the value of this attribute is 1, this attribute automatically unmounts VxFS snapshots when the file system is unmounted. If the value of this attribute is 0, and snapshots are mounted, the resource cannot be brought offline. In this case, failover does not occur. Type and dimension: integer-scalar Default: 0
CkptUmount	If the value of this attribute is 1, this attribute automatically unmounts VxFS checkpoints when file system is unmounted. If the value of this attribute is 0, and checkpoints are mounted, then failover does not occur. Type and dimension: integer-scalar Default: 1
ContainerName	Non-global zone support for Solaris 10 and above. Defines the name of the non-global zone. See the <i>Veritas Cluster Server User's Guide</i> for more information. Type and dimension: string-scalar Example: <code>zone1</code>
ContainerType	Do not change. For internal use only.

Table 2-6 Optional attributes

Optional attribute	Description
SecondLevelMonitor	This attribute is only applicable for an NFS client mount. It executes the <code>df -k</code> command for the NFS mounted file system and detects network outage. If the value of this attribute is 1, this attribute enables detailed monitoring of an NFS mounted file system. Type and dimension: boolean-scalar Default: 0
SecondLevelTimeout	This attribute is only applicable for an NFS client mount. This attribute is the timeout (in seconds) for the SecondLevelMonitor attribute that you try to request. The actual timeout value can be much smaller. This setting depends on how much time remains before it exceeds the MonitorTimeout interval. Type and dimension: integer-scalar Default: 30

Resource type definition

```
type Mount (  
  static keylist SupportedActions = { "mountpoint.vfd",  
    "mounted.vfd", "vxfslic.vfd", "chgmtlock", "mountentry.vfd" }  
  static str ArgList[] = { MountPoint, BlockDevice, FSType,  
    MountOpt, FsckOpt, SnapUmount, CkptUmount, SecondLevelMonitor,  
    SecondLevelTimeout, VxFSMountLock, ContainerName }  
  static str ContainerType = Zone  
  str MountPoint  
  str BlockDevice  
  str FSType  
  str MountOpt  
  str FsckOpt  
  int SnapUmount  
  int CkptUmount = 1  
  boolean SecondLevelMonitor = 0  
  int SecondLevelTimeout = 30  
  int VxFSMountLock = 1  
  str ContainerName  
)
```

Mount agent notes

The Mount agent has the following notes:

- [“High availability fire drill”](#) on page 47
- [“VxFS file system lock”](#) on page 47
- [“ZFS file system and pool creation example”](#) on page 47

High availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node; discrepancies that might prevent a service group from going online on a specific node. For Mount resources, the high availability drill performs the following, it:

- Checks if the specified mount point directory exists
- Checks if the mount point directory is already used
- Checks for valid Veritas (VxFS) file system licenses
- Checks if the mount point exists in the `/etc/filesystems` file

For more information about using the high availability fire drill see the *Veritas Cluster Server User's Guide*.

VxFS file system lock

If the mount option in the mount table output has the option `mntlock="key"`, then it is locked with the key `"key"`. To verify if mount locking is in use and has the value of `"key"`, run the `mount` command and review its output.

```
# mount
```

If the VxFS file system has `mntlock="key"` in its mount options, then unmounting the file system fails.

You can unlock the file system with the `fsadm` command and then unmount it. To unlock a locked mount, run the following command where `"key"` is the lock identifier and `mount_point_name` is the file system mount point.

```
# /opt/VRTS/bin/fsadm -o mntunlock="key" mount_point_name
```

To unmount a file system mounted with locking, run the `vxumount` command with the option `mntunlock="key"`, for example:

```
# /opt/VRTS/bin/vxumount -o mntunlock="key" mount_point_name
```

ZFS file system and pool creation example

If you want to use the Mount resource to monitor the ZFS file system, perform the following steps.

Create the tank storage pool and file system on the disk device c1t0d0 for example.

```
# zpool create tank c1t0d0
```

Create the home file system in tank.

```
# zfs create tank/home
```

Set the value of the MountPoint attribute to legacy.

```
# zfs set mountpoint=legacy tank/home
```

Set the Mount agent's attributes. The following is an example of this configuration's main.cf file.

```
Mount m1 (  
    MountPoint = "/mpl"  
    BlockDevice = "tank/home"  
    FSType = zfs  
    MountOpt = rw  
    FsckOpt = "-n"  
)
```

Sample configurations

VxFS configuration example

```
Mount mnt-fs1 (  
    MountPoint= "/mnt1"  
    BlockDevice = "/dev/vx/dsk/mnt-dg1/mnt-vol1"  
    FSType = "vxfs"  
    FsckOpt = "-n"  
    MountOpt = "rw"  
)
```

Zpool agent

The Zpool agent monitors ZFS storage pools. It exports ZFS storage pools (which reside on shared storage) from one node. It then imports the pool onto another node as required.

ZFS's automount feature mounts all its file systems by setting the mountpoint property to something other than legacy. To find the value of the mountpoint property, use the `zfs get` command. For example, from the command line for the tank mountpoint, enter:

```
# zfs get mountpoint tank
NAME          PROPERTY      VALUE          SOURCE
tank          mountpoint    /tank          default
```

As another example, to find the value of the mountpoint property for the legacypool storage pool, enter:

```
# zfs get mountpoint legacypool
NAME          PROPERTY      VALUE          SOURCE
tank          mountpoint    legacy         default
```

The Zpool agent checks this property, and checks the `ChkZFSMounts` attribute to decide whether the mounted file system should be checked in the Zpool agent or not.

When the value of the mountpoint property is one of the following:

- If the value of the mountpoint property is something other than legacy, the agent checks the mount status of the ZFS file systems.
- If the value of the mountpoint property is legacy, then it does not check the file system mount status. The agent assumes that you plan to use Mount resources to manage and monitor the ZFS file systems.

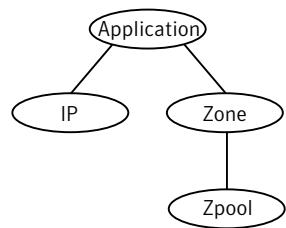
Limitations

The agent does not support the use of logical volumes in ZFS. If ZFS logical volumes are in use in the pool, the pool cannot be exported, even with the `-f` option. Sun does not recommend the use of logical volumes in ZFS due to performance and reliability issues.

Dependencies

No dependencies exist for the Zpool resource for a pool that has a non-legacy value for its mountpoint property.

Figure 2-6 Sample service group for a Zpool resource



Agent functions

Online	Imports the ZFS storage pool.
Offline	Exports the ZFS storage pool.
Monitor	Check the online status of the ZFS pool. If the mountpoint property of the ZFS file system is set and its value is not legacy, and the attribute ChkZFSMounts is enabled, then the agent checks if all the ZFS file systems under the same ZFS storage pool are mounted. If the ZFS pool contains a ZFS file system that a non-global zone uses, then you need to import the pool before the zone boots up. After the zone boots up, if the mountpoint property for this ZFS file system that the non-global zone uses is not set to legacy, it mounts after the zone boots up. If you have enabled the ChkZFSMounts in the Zpool resource, delay the check inside the Monitor agent function because the zone resource is not up yet, and the file systems are not mounted until the zone boots up. The Zone resource depends on the Zpool resource for the non-global zone scenario. In this case, you need to provide the ZoneResName attribute, which indicates the name of the Zone resource. When the Zone resource is in an ONLINE state, then ChkZFSMounts starts to check the mount status of the ZFS file system pool that the non-global zone uses.
Clean	Exports the ZFS storage pool forcefully.

State definitions

ONLINE	Reports an ONLINE state when the ZFS file systems that share a common storage pool are mounted, and the <code>zpool list -H -o health \$Poolname</code> indicates if the pool is online or not.
OFFLINE	Reports an OFFLINE state when all of the ZFS file systems that share a common storage pool are unmounted. It also reports an OFFLINE state when the <code>zpool list -H -o health \$Poolname</code> command's status indicates that the pool is offline.
UNKNOWN	Reports an UNKNOWN state in the following situations: ■ If the status of the storage pool is unavailable or faulted. ■ If the storage pool is online but the path of the mounted file system does not match the path that is specified in the <code>AltRootPath</code> attribute of this agent.

Attributes

Table 2-7 Required attributes

Required attribute	Description
PoolName	The name of the ZFS storage pool name. Type and dimension: string-scalar Default: n/a Example: tank

Table 2-7 Required attributes

Required attribute	Description
AltRootPath	Provides the alternate root path that is necessary to prevent the /etc/zfs/zpool.cache file from being populated. Supplying this value keeps a node from importing the ZFS storage pool automatically when it restarts after a crash. Not importing the ZFS storage prevents concurrency violations and file system corruption. If you do not provide a value for the AltRootPath attribute, VCS sets the \$AltRootPath to "/" as a workaround. This workaround makes sure that the ZFS command <code>zpool import -R \$AltRootPath \$PoolName</code> does not populate the zpool.cache file. Type and dimension: string-scalar Default: / Example: /mnt
ChkZFSMounts	The ChkZFSMounts attribute enables the check to determine whether all the file systems are properly mounted for that ZFS storage pool when the mountpoint property is not set to legacy. The default value is enabled (set to 1). Type and dimension: boolean-scalar Default: 1
ZoneResName	Use the ZoneResName attribute when a non-global zone needs the Zpool resource. In this case, supply the ZoneResName attribute with the name of the Zone resource. Type and dimension: string-scalar Default: n/a Example: zone1

Resource type definition

```
type Zpool (
    static str ArgList[] = { PoolName, AltRootPath, ChkZFMounts,
        ZoneResName, "ZoneResName:State" }
    str PoolName
    str AltRootPath
    boolean ChkZFMounts = 1
    str ZoneResName
)
```

Sample configurations

A main.cf example that shows the Zpool agent configuration.

```
include "types.cf"

cluster clus1 (
    UserNames = { admin = dqrJqlQnrMrrPzrLqo,
        z_zone_res = dOMoOTnNMlMSlVPnOT,
        z_dummy_res = bIJbIDiFJeJJhRjDIG }
    Administrators = { admin }
)

system sysA (
)

system sysB (
)

group tstzfs (
    SystemList = { sysA = 0, sysB = 1 }
    AutoStartList = { sysA, sysB }
    Administrators = { z_zone_res }
)

Zone zone_res (
    ZoneName = z1
)

Zpool legacy_res (
    PoolName = legacypool
    ZoneResName = zone_res
)

Zpool tstzone_res (
    PoolName = tstzonepool
    ZoneResName = zone_res
)

Zpool zpool_res (
```

```
PoolName = tank
AltRootPath = "/mytank"
)

Application custom_app (
  StartProgram = "/mytank/tank/startapp"
  StopProgram = "/mytank/tank/stopapp"
  MonitorProcesses = { "/mytank/tank/mondaemon" }
)

custom_app requires zpool_res
zone_res requires legacy_res
zone_res requires tstzone_res
```

Network agents

This chapter contains the following:

- [“About the network agents”](#) on page 55
- [“IP agent”](#) on page 58
- [“NIC agent”](#) on page 64
- [“IPMultiNIC agent”](#) on page 68
- [“MultiNICA agent”](#) on page 72
- [“About the IPMultiNICB and MultiNICB agents”](#) on page 79
- [“IPMultiNICB agent”](#) on page 80
- [“MultiNICB agent”](#) on page 85
- [“DNS agent”](#) on page 98

About the network agents

Use network agents to provide high availability for networking resources.

Agent comparisons

IP and NIC agents

The IP and NIC agents:

- Monitor a single NIC

IPMultiNIC and MultiNICA agents

The IPMultiNIC and MultiNICA agents:

- Monitor single or multiple NICs
- Check the backup NICs at fail over
- Use the original base IP address when failing over
- Provide slower failover compared to MultiNICB but can function with fewer IP addresses
- Have only one active NIC at a time

IPMultiNICB and MultiNICB agents

The IPMultiNICB and MultiNICB agents:

- Monitor single or multiple NICs
- Check the backup NICs as soon as it comes up
- Require a pre-assigned base IP address for each NIC
- Do not fail over the original base IP address
- Provide faster fail over compared to MultiNICA but require more IP addresses
- Have more than one active NIC at a time

802.1Q trunking

The IP/NIC, IPMultiNIC/MultiNICA, and IPMultiNICB/MultiNICB agents support 802.1Q trunking.

The IP/NIC, IPMultiNIC/MultiNICA, and IPMultiNICB/MultiNICB agents support 802.1Q trunking on Solaris 8, 9 and 10. However, on Solaris 8, only "ce" interfaces can be configured as VLAN interfaces. This is a Sun restriction.

On Solaris 9, the IPMultiNICB and MultiNICB agents works only if Sun patch 116670-04 is installed on the system. No patch is required for the IP and NIC agents and the IPMultiNIC and MultiNICA agents

On Solaris 9 and 10, VLAN is not supported on the Fast Ethernet interfaces. (eg: hme/qfe interfaces).

You need to specify the VLAN interfaces, for example: bge20001 , bge30001, as the base interfaces in the device list in the main.cf file. You also must make sure that the IP addresses that are assigned to the interfaces of a particular VLAN are in the same subnet.

IP agent

The IP agent manages the process of configuring a virtual IP address and its subnet mask on an interface. The virtual IP address must not be in use. You can use this agent when you want to monitor a single IP address on a single adapter.

High availability fire drill

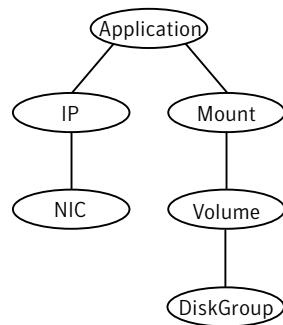
The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node. These discrepancies might prevent a service group from going online on a specific node. For IP resources, the high availability fire drill checks for the existence of a route to the IP from the specified NIC.

For more information about using the high availability fire drill see the *Veritas Cluster Server User's Guide*.

Dependencies

IP resources depend on NIC resources.

Figure 3-1 Sample service group for an IP resource



Agent functions

Online	Configures the IP address to the NIC. Checks if another system is using the IP address. Uses the <code>ifconfig</code> command to set the IP address on a unique alias on the interface.
Offline	Brings down the IP address that is specified in the Address attribute.
Monitor	Monitors the interface to test if the IP address that is associated with the interface is alive.
Clean	Brings down the IP address that is associated with the specified interface.

State definitions

ONLINE	Indicates that the device is up and the specified IP address is assigned to the device.
OFFLINE	Indicates that the device is down or the specified IP address is not assigned to the device.
UNKNOWN	Indicates that the agent could not determine the state of the resource or that the resource attributes are invalid.

Attributes

Table 3-1 Required attributes

Required attribute	Description
Address	A virtual IP address that is associated with the interface. Note that the address you specify must not be the same as the configured physical IP address, but should be on the same network. Type and dimension: string-scalar Example: "192.203.47.61"
Device	The name of the NIC device that is associated with the IP address. Requires the device name without an alias. Type and dimension: string-scalar Example: "le0"

Table 3-2 Optional attributes

Optional attribute	Description
ArpDelay	The number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about this IP address. Type and dimension: integer-scalar Default: 1
ContainerName	Non-global zone support for Solaris 10 and later. Defines the name of the non-global zone. See the <i>Veritas Cluster Server User's Guide</i> for more information. Type and dimension: string-scalar Example: zone1

Table 3-2 Optional attributes

Optional attribute	Description
IfconfigTwice	Causes an IP address to be configured twice using an ifconfig up-down-up sequence. Increases the probability of gratuitous ARP requests (generated by ifconfig up) to reach clients. Type and dimension: integer-scalar Default: 0
NetMask	The subnet mask that is associated with the IP address of the resource. Specify the value of the netmask in decimal (base 10) or hexadecimal (base 16). Symantec recommends that you specify a netmask for each virtual interface. Type and dimension: string-scalar Default: + If you do not specify the netmask in the ifconfig command, the agent uses a default netmask that is based on the contents of the /etc/netmasks path for a given address range. Example: "255.255.248.0"
Options	Options for the ifconfig command. Type and dimension: string-scalar Example: "trailers"

Table 3-2Optional attributes

Optional attribute	Description
RouteOptions	Specifies the routing options that are passed to the <code>route add</code> command when the agent configures an interface. The <code>RouteOptions</code> attribute value is generally formed like this: "<i>destination gateway metric</i>". For details about the <code>route</code> command, refer to the man page for your operating system. When the value of this string is null, the agent does not add routes. Type and dimension: string-scalar Example: "192.100.201.0 192.100.13.7" In this example, the agent executes the "<code>route add 192.100.201.0 192.100.13.7</code>" command when it configures an interface.

Resource type definition

```
type IP (  
  static keylist SupportedActions = { "device.vfd", "route.vfd" }  
  static str ArgList[] = { Device, Address, NetMask, Options,  
    ArpDelay, IfconfigTwice, RouteOptions, ContainerName }  
  str Device  
  str Address  
  str NetMask  
  str Options  
  int ArpDelay = 1  
  int IfconfigTwice  
  str RouteOptions  
  str ContainerName  
)
```

Sample configurations

Configuration 1

```
IP          IP_192_203_47_61 (
  Device = le0
  Address = "192.203.47.61"
)
```

NetMask in decimal (base 10)

```
IP          IP_192_203_47_61 (
  Device = le0
  Address = "192.203.47.61"
  NetMask = "255.255.248.0"
)
```

Configuration of NetMask in hexadecimal (base 16)

```
IP          IP_192_203_47_61 (
  Device = le0
  Address = "192.203.47.61"
  NetMask = "0xfffff800"
)
```

NIC agent

The NIC agent monitors the configured NIC. If a network link fails, or if a problem arises with the NIC, the resource is marked `FAULTED`. You can use the agent to make a single IP address on a single adapter highly available or to monitor it. This resource's Operation value is `OnOnly`.

The NIC listed in the Device attribute must have an administrative IP address. The administrative IP address is the default IP address that is assigned to the physical interface of a host on a network. This agent does not configure network routes or administrative IP addresses.

Before you use this agent:

- Verify that the NIC has the correct administrative IP address and subnet mask.
- Verify that the NIC does not have built-in failover support. If it does, disable it.

High availability fire drill

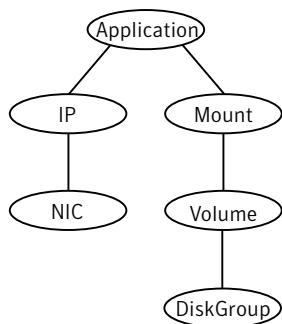
The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node. These discrepancies might prevent a service group from going online on a specific node. For NIC resources, the high availability fire drill checks for the existence of the NIC on the host.

For more information about using the high availability fire drill see the *Veritas Cluster Server User's Guide*.

Dependencies

No child dependencies exist for this resource.

Figure 3-2 Sample service group for a NIC resource



Agent functions

Monitor	Tests the network card and network link. Pings the network hosts or broadcast address of the interface to generate traffic on the network. Counts the number of packets passing through the device before and after the address is pinged. If the count decreases or remains the same, the resource is marked <code>FAULTED</code>. If the <code>NetworkHosts</code> list is empty, or the ping test fails, the agent sends a ping to the device's broadcast address to generate network traffic. The agent checks for any response to the broadcast request. If there is no reply to the broadcast ping, the resource faults.
---------	--

State definitions

ONLINE	Indicates that the NIC resource is working.
FAULTED	Indicates that the NIC has failed.
UNKNOWN	Indicates the agent cannot determine the interface state. It may be due to an incorrect configuration.

Attributes

Table 3-3 Required attributes

Required attribute	Description
Device	Name of the NIC that you want to monitor. Type and dimension: string-scalar Example: "le0"

Table 3-4 Optional attributes

Optional attribute	Description
NetworkHosts	List of hosts on the network that are pinged to determine if the network connection is alive. You can use this attribute to help to save network capacity and reduce monitor time. Symantec recommends that you use the outgoing gateway routers for this value. Enter the IP address of the host, instead of the host name, to prevent the monitor from timing out. DNS causes the ping to hang. If more than one network host is listed, the monitor returns ONLINE if at least one of the hosts is alive. If you do not specify network hosts, the monitor tests the NIC by sending pings to the broadcast address on the NIC. Type and dimension: string-vector Example: "166.96.15.22", "166.97.1.2"
NetworkType	Type of network. VCS supports only Ethernet. Type and dimension: string-scalar Default: "ether"

Table 3-4 Optional attributes

Optional attribute	Description
PingOptimize	Number of monitor cycles to detect if a configured interface is inactive. Use PingOptimize when you have not specified the NetworkHosts attribute. A value of 1 optimizes broadcast pings and requires two monitor cycles. A value of 0 performs a broadcast ping during each monitor cycle and detects the inactive interface within the cycle. Type and dimension: integer-scalar Default: 1

Resource type definition

```
type NIC (  
    static keylist SupportedActions = { "device.vfd" }  
    static str ArgList[] = { Device, NetworkType, PingOptimize,  
        NetworkHosts}  
    static int OfflineMonitorInterval = 60  
    static str Operations = None  
    str Device  
    str NetworkType  
    int PingOptimize = 1  
    str NetworkHosts[]  
)
```

Sample configurations

Configuration without network hosts (using default ping mechanism)

```
NIC groupx_le0 (  
    Device = le0  
    PingOptimize = 1  
)
```

Configuration with network hosts

```
NIC groupx_le0 (  
    Device = le0  
    NetworkHosts = { "166.93.2.1", "166.99.1.2" }  
)
```

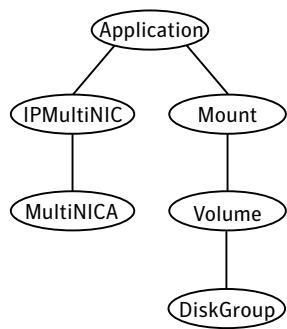
IPMultiNIC agent

The IPMultiNIC agent manages the virtual IP address that is configured as an alias on one interface of a MultiNICA resource. If the interface faults, the agent works with the MultiNICA resource to fail over to a backup NIC. If multiple service groups have IPMultiNICs associated with the same MultiNICA resource, only one group has the MultiNICA resource. The other groups have Proxy resources pointing to it. You can use this agent for IP addresses on multiple-adaptor systems.

Dependencies

IPMultiNIC resources depend on MultiNICA resources.Can depend on Zone resources.

Figure 3-3 Sample service group for an IPMultiNIC resource



Agent functions

Online	Configures a virtual IP address on one interface of the MultiNICA resource.
Offline	Removes the virtual IP address from one interface of the MultiNICA resource.
Monitor	Checks if the virtual IP address is configured on one interface of the MultiNICA resource.
Clean	Removes the virtual IP address from one interface of the MultiNICB resource.

State definitions

ONLINE	Indicates that the specified IP address is assigned to the device.
OFFLINE	Indicates that the specified IP address is not assigned to the device.
UNKNOWN	Indicates that the agent can not determine the state of the resource. This state may be due to an incorrect configuration.

Attributes

Table 3-5 Required attributes

Required attribute	Description
Address	Virtual IP address assigned to the active NIC. Type and dimension: string-scalar Example: "10.128.10.14"
MultiNICResName	Name of the associated MultiNICA resource that determines the active NIC. Type and dimension: string-scalar Example: "mnic"

Table 3-6 Optional attributes

Optional attribute	Description
ContainerName	Non-global zone support for Solaris 10 and above. Defines the name of the non-global zone. Type and dimension: string-scalar Example: "zone1"

Table 3-6 Optional attributes

Optional attribute	Description
IfconfigTwice	Causes an IP address to be configured twice using an <code>ifconfig up-down-up</code> sequence. Increases the probability of gratuitous ARP requests (generated by <code>ifconfig up</code>) to reach clients. Type and dimension: integer-scalar
NetMask	The netmask that is associated with the IP address of the resource. Specify the value of the netmask in decimal (base 10) or hexadecimal (base 16). Symantec recommends that you specify a netmask for each virtual interface. Type and dimension: string-scalar Default: + If you do not specify the netmask in the <code>ifconfig</code> command, the agent uses a default netmask. The default netmask is based on the contents of the <code>/etc/netmasks</code> for a given address range. Example: "255.255.248.0"
Options	The <code>ifconfig</code> command options for the virtual IP address. Type and dimension: string-scalar Example: "trailers"

Note: On Solaris systems, Symantec recommends that you set the `RestartLimit` for IPMultiNIC resources to a greater-than-zero value. Setting this value helps to prevent the spurious faulting of IPMultiNIC resources during local failovers of MultiNICA. A local failover is an interface-to- interface failover of MultiNICA. See the *VCS User's Guide* for more information.

Resource type definition

```
type IPMultiNIC (
    static str ArgList[] = { "MultiNICResName:Device", Address,
                             NetMask, "MultiNICResName:ArpDelay", Options,
                             "MultiNICResName:Probed", MultiNICResName, IfconfigTwice,
                             ContainerName }
    static int MonitorTimeout = 120
    str Address
    str NetMask
    str Options
    str MultiNICResName
    int IfconfigTwice
    str ContainerName
)
```

Sample configuration: IPMultiNIC and MultiNICA

Refer to the MultiNICA agent for more information.

```
group grp1 (
    SystemList = { sysa, sysb }
    AutoStartList = { sysa }
)
MultiNICA mnic (
    Device@sysa = { le0 = "10.128.8.42", qfe3 = "10.128.8.42" }
    Device@sysb = { le0 = "10.128.8.43", qfe3 = "10.128.8.43" }
    NetMask = "255.255.255.0"
    ARPDelay = 5
    Options = "trailers"
)
IPMultiNIC ip1 (
    Address = "10.128.10.14"
    NetMask = "255.255.255.0"
    MultiNICResName = mnic
    Options = "trailers"
)
ip1 requires mnic
group grp2 (
    SystemList = { sysa, sysb }
    AutoStartList = { sysa }
)
    IPMultiNIC ip2 (
        Address = "10.128.9.4"
        NetMask = "255.255.255.0"
        MultiNICResName = mnic
        Options = "trailers"
    )
    Proxy proxy (
        TargetResName = mnic
    )
ip2 requires proxy
```

MultiNICA agent

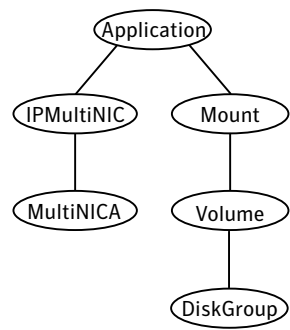
The MultiNICA represents a set of network interfaces and provides failover capabilities between them. You can use the agent to make IP addresses on multiple-adapter systems highly available or to monitor them. Each interface in a MultiNICA resource has a base IP address. You can use one base IP address for all NICs, or you can specify a different IP address for use with each NIC. The MultiNICA agent configures one interface at a time. If it does not detect activity on the configured interface, it configures a new interface and migrates IP aliases to it.

If an interface is associated with a MultiNICA resource, do not associate it with any other MultiNICA, MultiNICB, or NIC resource. If the same set of interfaces must be a part of multiple service groups, configure a MultiNICA resource in one of the service groups. Configure the Proxy resources that point to the MultiNICA resource in the other service groups.

Dependencies

No dependencies exist for the MultiNICA resource.

Figure 3-4 Sample service group for a MultiNICA resource



Agent function

Monitor	Checks the status of the active interface. If the agent detects a failure, it tries to migrate the IP addresses that are configured on that interface. If possible, it tries to migrate the addresses to the next available interface that is configured in the Device attribute.
---------	---

State definitions

ONLINE	Indicates that one or more of the network interfaces listed in the Device attribute of the resource is in working condition.
OFFLINE	Indicates that all of the network interfaces listed in the Device attribute failed.
UNKNOWN	Indicates that the agent cannot determine the state of the network interfaces that are specified in the Device attribute. This state may be due to incorrect configuration.

Attributes

Table 3-7 Required attributes

Required attribute	Description
Device	List of interfaces and their base IP addresses. Type and dimension: string-association Example: { le0 = "10.128.8.42", qfe3 = "10.128.8.42" }

Table 3-8 Optional attributes

Optional attribute	Description
ArpDelay	Number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about the base IP address. Type and dimension: integer-scalar Default: 1

Table 3-8 Optional attributes

Optional attribute	Description
HandshakeInterval	Computes the maximum number of tries the agent makes either to: ■ ping a host (listed in the NetworkHosts attribute) when it fails over to a new NIC, or ■ to ping the default broadcast address (depending on the attribute configured) when it fails over to a new NIC. If the value of the RetestInterval attribute is five (default), each try takes about 10 seconds. To prevent spurious failovers, the agent must try to contact a host on the network several times before it marks a NIC as FAULTED. Increased values result in longer failover times, whether between the NICs or from system to system in the case of FAULTED NICs. Type and dimension: integer-scalar Default: 20 This value is the equivalent to two tries (20/10).
IfconfigTwice	Causes an IP address to be configured twice, using an <code>ifconfig up-down-up</code> sequence. Increases the probability of gratuitous ARP requests (caused by <code>ifconfig up</code>) to reach clients. Type and dimension: integer-scalar
NetMask	Netmask for the base IP address. Specify the value of NetMask in decimal (base 10) or hexadecimal (base 16). Note: Symantec recommends that you specify a netmask for each virtual interface. Type and dimension: string-scalar Default: + Example: "255.255.255.0"

Table 3-8 Optional attributes

Optional attribute	Description
NetworkHosts	The list of hosts on the network that are pinged to determine if the network connection is alive. Enter the IP address of the host, instead of the host name, to prevent the monitor from timing out—DNS causes the ping to hang. If this attribute is unspecified, the monitor tests the NIC by pinging the broadcast address on the NIC. If more than one network host is listed, the monitor returns online if at least one of the hosts is alive. Type and dimension: string-vector Example: "128.93.2.1", "128.97.1.2"
Options	The <code>ifconfig</code> options for the base IP address. Type and dimension: string-scalar Example: "trailers"
PingOptimize	Number of monitor cycles to detect if the configured interface is inactive. A value of 1 optimizes broadcast pings and requires two monitor cycles. A value of 0 performs a broadcast ping each monitor cycle and detects the inactive interface within the cycle. Type and dimension: integer-scalar Default: 1
RetestInterval	Number of seconds to sleep between re-tests of a newly configured interface. A lower value results in faster local (interface-to-interface) failover. Type and dimension: integer-scalar Default: 5
RouteOptions	String to add a route when configuring an interface. Use only when configuring the local host as the default gateway. The string contains destination gateway metric. No routes are added if this string is set to NULL. Type and dimension: string-scalar Example: "default 166.98.16.103 0"

Resource type definition

```
type MultiNICA (
    static str ArgList[] = { Device, NetMask, ArpDelay,
        RetestInterval, Options, RouteOptions, PingOptimize,
        MonitorOnly, IfconfigTwice, HandshakeInterval, NetworkHosts }
    static int OfflineMonitorInterval = 60
    static int MonitorTimeout = 300
    static str Operations = None
    str Device{}
    str NetMask
    int ArpDelay = 1
    int RetestInterval = 5
    str Options
    str RouteOptions
    int PingOptimize = 1
    int IfconfigTwice
    int HandshakeInterval = 20
    str NetworkHosts[]
)
```

MultiNICA notes

- If all NICs configured in the Device attribute are down, the MultiNICA agent faults the resource after a two-three minute interval. This delay occurs because the MultiNICA agent tests the failed NIC several times before it marks the resource OFFLINE. Failover logs record a detailed description of the events.
- The engine log is in `/var/VRTSvcs/log/engine_A.log`.
- The MultiNICA agent supports only one active NIC on one IP subnet; the agent does not work with multiple active NICs on the same subnet.
 - On Solaris, for example, you have two active NICs, hme0 (10.128.2.5) and qfe0 (10.128.2.8). You configure a third NIC, qfe1, as the backup NIC to hme0. The agent does not fail over from hme0 to qfe1 because all ping tests are redirected through qfe0 on the same subnet. The redirect makes the MultiNICA monitor return an online status. Note that using `ping -i` does not enable the use of multiple active NICs.
- Before you start VCS, configure the primary NIC with the correct broadcast address and netmask.
 - Set the NIC here: `/etc/hostname.nic`
 - Set the netmask here: `/etc/netmask`

Using RouteOptions

The RouteOptions attribute is useful only when the default gateway is your own host.

For example, if the default gateway and hme0 are both set to 10.128.8.42, the output of the `netstat -rn` command resembles:

Destination	Gateway	Flags	Ref	Use	Interface
10.0.0.0	10.128.8.42	U	1	2408	hme0
224.0.0.0	10.128.8.42	U	1	0	hme0
default	10.128.8.42	UG	1	2402	hme0
127.0.0.1	127.0.0.1	UH	54	44249	lo0

If the RouteOptions attribute is not set and hme0 fails, the MultiNICA agent migrates the base IP address to another NIC (such as qfe0). The default route is no longer configured because it was associated with hme0. The display resembles:

Destination	Gateway	Flags	Ref	Use	Interface
10.0.0.0	10.128.8.42	U	1	2408	qfe0
224.0.0.0	10.128.8.42	U	1	0	qfe0
127.0.0.1	127.0.0.1	UH	54	44249	lo0

If the RouteOptions attribute defines the default route, the default route is reconfigured on the system. For example:

```
RouteOptions@sysa = "default 10.128.8.42 0"  
RouteOptions@sysb = "default 10.128.8.43 0"
```

Sample configurations

MultiNICA and IPMultiNIC

In the following example, two nodes, sysa and sysb, each have a pair of network interfaces, le0 and qfe3. In this example, the two interfaces, le0 and qfe3, have the same base, or physical, IP address. Note the lines beginning Device@sysa and Device@sysb; the use of different physical addresses shows how to localize an attribute for a particular host.

The MultiNICA resource fails over only the physical IP address to the backup NIC during a failure. The IPMultiNIC agent configures the logical IP addresses. The resources ip1 and ip2, shown in the following example, have the Address attribute that contains the logical IP address. If a NIC fails on sysa, the physical IP address and the two logical IP addresses fails over from le0 to qfe3. If qfe3 fails, the address fails back to le0 if le0 is reconnected.

However, if both the NICs on sysa are disconnected, the MultiNICA and IPMultiNIC resources work in tandem to fault the group on sysa. The entire group now fails over to sysb.

If you have more than one group using the MultiNICA resource, the second group can use a Proxy resource. The Proxy resource points to the MultiNICA resource in the first group. The Proxy resource prevents redundant monitoring of the NICs on the same system. The IPMultiNIC resource is always made dependent on the MultiNICA resource. See the IPMultiNIC agent for more information.

```
group grp1 (
    SystemList = { sysa, sysb }
    AutoStartList = { sysa }
)
MultiNICA mnic (
    Device@sysa = { le0 = "10.128.8.42", qfe3 = "10.128.8.42" }
    Device@sysb = { le0 = "10.128.8.43", qfe3 = "10.128.8.43" }
    NetMask = "255.255.255.0"
    ArpDelay = 5
    Options = "trailers"
)

IPMultiNIC ip1 (
    Address = "10.128.10.14"
    NetMask = "255.255.255.0"
    MultiNICResName = mnic
    Options = "trailers"
)
```

ip1 requires mnic

```
group grp2 (
    SystemList = { sysa, sysb }
    AutoStartList = { sysa }
)
IPMultiNIC ip2 (
    Address = "10.128.9.4"
    NetMask = "255.255.255.0"
    MultiNICResName = mnic
    Options = "trailers"
)
Proxy proxy (
    TargetResName = mnic
)
```

ip2 requires proxy

About the IPMultiNICB and MultiNICB agents

The IPMultiNICB and the MultiNICB agents can handle multiple NIC connections. Due to differences in the way that each platform handles its networking connections, these agents vary in design between platforms.

Checklist to ensure the proper operation of MultiNICB

For the MultiNICB agent to function properly, you must satisfy each item in the following list:

- Each interface must have a unique MAC address.
- A MultiNICB resource controls all the interfaces on one IP subnet.
- At boot time, you must configure and connect all the interfaces that are under the MultiNICB resource and give them test IP addresses.
- All test IP addresses for the MultiNICB resource must belong to the same subnet as the virtual IP address.
- Reserve the base IP addresses, which the agent uses to test the link status, for use by the agent. These IP addresses do not get failed over.
- The IgnoreLinkStatus attribute is set to 1 (default) when using trunked interfaces.
- If you specify the NetworkHosts attribute, then that host must be on the same subnet as the other IP addresses for the MultiNICB resource.
- Test IP addresses have "nofailover" and "deprecated" flags set at boot time.
- `/etc/default/mpathd/` has `TRACK_INTERFACES_ONLY_WITH_GROUPS=yes`.
- If you are not using Solaris `in.mpathd`, all MultiNICB resources on the system have the UseMpathd attribute set to 0 (default). You cannot run `in.mpathd` on this system.
- If you are using Solaris `in.mpathd`, all MultiNICB resources on the system have the UseMpathd attribute set to 1.

IPMultiNICB agent

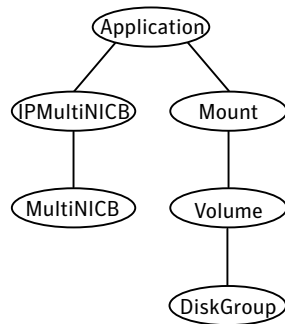
The IPMultiNICB agent works with the MultiNICB agent. The agent configures and manages virtual IP addresses (IP aliases) on an active network device that the MultiNICB resource specifies. When the MultiNICB agent reports a particular interface as failed, the IPMultiNICB agent moves the IP address to the next active interface. You can use this agent for IP addresses on multiple-adaptor systems.

If multiple service groups have IPMultiNICB resources associated with the same MultiNICB resource, only one group should have a MultiNICB resource. The other groups should have a proxy resource pointing to the MultiNICB resource. For the MultiNICB and IPMultiNICB agents, VCS supports Sun trunking.

Dependencies

IPMultiNICB resources depend on MultiNICB resources.

Figure 3-5 Sample service group for an IPMultiNICB resource



Requirements for IPMultiNICB

The following conditions must exist for the IPMultiNICB agent to function correctly:

- The MultiNICB agent must be running to inform the IPMultiNICB agent of the available interfaces.
- Only one IPMultiNICB agent can control each logical IP address.

Agent functions

Online	Finds a working interface with the appropriate interface alias or interface name, and configures the logical IP address on it.
Offline	Removes the logical IP address.
Clean	Removes the logical IP address.
Monitor	If the logical IP address is not configured as an alias on one of the working interfaces under a corresponding MultiNICB resource, monitor returns OFFLINE. If the current interface fails, the agent fails over the logical IP address. It fails over the logical IP address to the next available working interface that is within the MultiNICB resource on the same node. If no working interfaces are available then monitor returns OFFLINE.

State definitions

ONLINE	Indicates that an IP address on one of the working network interfaces of the resource is up. The IP address is specified in the Address attribute. The resource is specified in the BaseResName attribute.
OFFLINE	Indicates that an IP address on one of the working network interfaces of the resource is up. The IP address is specified in the Address attribute. The resource is specified in the BaseResName attribute.
UNKNOWN	Indicates that the agent cannot determine the status of the virtual IP address that is specified in the Address attribute.
FAULTED	The IP address could not be brought online, usually due to all NICs in the MultiNICB resource faulting.

Attributes

Table 3-9 Required attributes

Required attribute	Description
Address	The logical IP address that the IPMultiNICB resource must handle. This IP address must be different than the base or test IP addresses in the MultiNICB resource. Type and dimension: string-scalar Example: "10.112.10.15"
BaseResName	Name of MultiNICB resource from which the IPMultiNICB resource gets a list of working interfaces. The logical IP address is placed on the physical interfaces according to the device number information. Create the BaseResName attribute in this format: <i>service_group_name.mnicb_resource_name</i> This service group is where the mnicb_resource resides. Type and dimension: string-scalar Example: "gnic_n"
NetMask	The netmask that is associated with the logical IP address. Type and dimension: string-scalar Example: "255.255.255.0"

Table 3-10 Optional attributes

Optional attribute	Description
ContainerName	Non-global zone support for Solaris 10 and later. Defines the name of the non-global zone. Type and dimension: string-scalar Example: "zone1"

Table 3-10 Optional attributes

Optional attribute	Description
DeviceChoice	Indicates the preferred NIC where you want to bring the logical IP address online. Specify the device name or NIC alias as determined in the Device attribute of the MultiNICB resource. Type and dimension: string-scalar Default: 0 Examples: "qfe0" and "1"
RouteOptions	Specifies the routing options that are passed to the <code>route add</code> command when the agent configures an interface. The RouteOptions attribute value is generally formed like this: "<i>destination gateway metric</i>". For details about the <code>route</code> command, refer to the man page for your operating system. When the value of this string is null, the agent does not add routes. Type and dimension: string-scalar Example: "192.100.201.0 192.100.13.7" In this example, the agent executes the "<code>route add 192.100.201.0 192.100.13.7</code>" command when it configures an interface.

Note: The value of the ToleranceLimit static attribute is 1. A value of 1 avoids spurious agent faults in the Multipathing mode while Sun's mpathd daemon migrates the IP address from one interface to another.

Due to the change in the ToleranceLimit attribute, the value of the MonitorInterval static attribute is now 30 seconds. The 30-second value means that the agent tries to online the resource twice a minute. This value ensures that the overall fault detection time is still 60 seconds.

Resource type definition

```
type IPMultiNICB (
    static int ToleranceLimit = 1
    static int MonitorInterval = 30
    static str ArgList[] = { BaseResName, Address, NetMask,
        DeviceChoice, RouteOptions, ContainerName }
    str BaseResName
    str Address
    str NetMask
    str DeviceChoice = 0
    str RouteOptions
    str ContainerName
)
```

Manually migrating a logical IP address Solaris

Use the `haipswitch` command to migrate the logical IP address from one interface to another.

This command shows the status of the interfaces for the specified MultiNICB resource:

```
# haipswitch -s MultiNICB_resname
```

In the following example, the command checks that both the *from* and *to* interfaces are associated with the specified MultiNICB resource. The command also checks if the *to* interface works. If the interface does not work, the command aborts the operation. It then removes the IP address on the *from* logical interface and configures the IP address on the *to* logical interface. It finally erases any previous failover information that is created by MultiNICB for this logical IP address.

```
# haipswitch MultiNICB_resname IPMultiNICB_resname ip addr \
    netmask from to
```

Sample configurations

Other sample configurations for IPMultiNICB and MultiNICB

Refer to the sample configurations in the MultiNICB agent.

MultiNICB agent

The MultiNICB works with the IPMultiNICB agent. Allows IP addresses to fail over to multiple NICs on the same system before VCS tries to fail over to another system. You can use the agent to make IP addresses on multiple-adaptor systems highly available or to monitor them.

When you use the MultiNICB agent, you must configure the NICs before putting them under the agent's control. You must configure all the NICs in a single MultiNICB resource with the IP addresses that are in the same subnet.

If multiple service groups have IPMultiNICB resources associated with the same MultiNICB resource, only one group should have the MultiNICB resource. The other groups can have a proxy resource pointing to it.

For the MultiNICB and IPMultiNICB agents, VCS supports Sun trunking.

Base and Multipathing modes

You can use the MultiNICB agent in one of two modes. They are:

- Base mode
- Multipathing mode

See “[Solaris operating modes: Base and Multipathing](#)” on page 93.

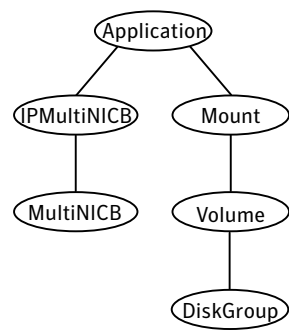
Sun trunking

You can configure MultiNICB for use with a single trunk head or multiple trunk heads. You need to set the value of the IgnoreLinkStatus attribute to 1. You must also ensure that all interfaces that belong to the same MultiNICB resource are in the same subnet.

Dependencies

No dependencies exist for the MultiNICB resource.

Figure 3-6 Sample service group for a MultiNICB resource



Agent functions

Open	Allocates an internal structure to store information about the resource.
Close	Frees the internal structure that is used to store information about the resource.
Monitor	Checks the status of each physical interface. Writes the status information to the export information file for IPMultiNICB resources to read it. Performs a failover. Performs failback if the value of the Failback attribute is 1.

State definitions

ONLINE	Indicates that one or more of the network interfaces listed in the Device attribute of the resource is in working condition.
UNKNOWN	Indicates that the MultiNICB resource is not configured correctly.
FAULTED	Indicates that all of the network interfaces listed in the Device attribute failed.

Attributes

Table 3-11 Required attributes

Required attribute	Description
Device	List of NICs that you want under MultiNICB control, and the aliases of those NICs. The IPMultiNICB agent uses the NIC aliases to configure IP addresses. The IPMultiNICB agent uses these interface aliases to determine the order of the interface on which to bring the IP addresses online. Type and dimension: string-association Examples: In this example, the MultiNICB agent uses interfaces qfe0, qfe1, and qfe2. The MultiNICB agent passes on the associated interface aliases 0, 2, and 3 to the IPMultiNICB agent. Device = { "qfe0" , "qfe4" } Device = { "qfe0" = 0, "qfe1" = 2, "qfe2" = 3 }

Optional attributes for Base and Mpathd modes

Table 3-12 Optional attributes for Base and Mpathd modes

Optional attribute	Description
DefaultRouter	This attribute is the IP address of the default router on the subnet. If you specify this value, the agent removes the default route when the resource goes offline. The agent adds the route back when the group returns online. You must specify this attribute if multiple IP subnets exist on one host. If you do not specify the value, the packets cannot be routed properly when the subnet corresponding to the first default route goes down. Type and dimension: string-scalar Default: 0.0.0.0 Example: "192.1.0.1"

Table 3-12 Optional attributes for Base and Mpathd modes

Optional attribute	Description
GroupName	The GroupName attribute is the name of the IPMP group that you want to assign to the interfaces under the control of the agent. The name's length should not exceed 31 bytes. Type and dimension: string-scalar Example: "IPMPgrp1"
MpathdCommand	This value is the path to the mpathd executable. Use MpathdCommand to kill or restart mpathd. See the UseMpathd attribute for details. Type and dimension: string-scalar Default: /usr/lib/inet/in.mpathd
UseMpathd	The legal values for this attribute are 0 and 1. All the MultiNICB resources on one system must have the same value for this attribute. See “Solaris operating modes: Base and Multipathing” on page 93. If the value of the attribute is 0, in.mpathd is automatically killed on that system. For more information about mpathd, refer to the Sun documentation. If the value of the attribute is 1, MultiNICB assumes that mpathd (in.mpathd) is running. This value restarts mpathd if it is not running already. Type and dimension: integer-scalar Default: 0

Optional attributes for Base mode

Table 3-13 Optional attributes for Base mode

Optional attribute	Description
Failback	If the value of the attribute is 1, the virtual IP addresses are failed back to the original physical interface whenever possible. A value of 0 disables this behavior. Type and dimension: integer-scalar Default: 0
IgnoreLinkStatus	If the value of the attribute is 1, the agent ignores the driver-reported interface status while testing the interfaces. If the value of the attribute is 0, the agent reports the interface status as DOWN if the driver-reported interface status indicates the DOWN state. Using interface status for link testing may considerably speed up failovers. When you use trunked interfaces, you must set the value of this attribute to 1. Otherwise set it to 0. Type and dimension: integer-scalar Default: 1

Table 3-13 Optional attributes for Base mode

Optional attribute	Description
LinkTestRatio	This attribute is the ratio of total monitor cycles to monitor cycles in which the agent tests the interfaces by sending packets. At all other times, the agent tests the link by checking the "link-status" as reported by the device driver. Checking the "link-status" is a faster way to check the interfaces, but only detects cable disconnection failures. If the value of the attribute is 1, packets are sent during every monitor cycle. If the value of the attribute is 0, packets are never sent during a monitor cycle. Type and dimension: integer-scalar Default: 1 Example: "3" In this example, if monitor entry-point invoking is numbered as 1, 2, 3, 4, 5, 6, ..., the actual packet send test is done at 3, 6, ... monitor entry-points. For LinkTestRatio=4, the packet send test is done at 4, 8, ... monitor agent functions.
NetworkHosts	List of host IP addresses on the IP subnet that are pinged to determine if the interfaces work. NetworkHosts only accepts IP addresses to avoid DNS lookup delays. The IP addresses must be directly present on the IP subnet of interfaces (the hosts must respond to ARP requests). If IP addresses are not provided, the hosts are automatically determined by sending a broadcast ping (unless the NoBroadcast attribute is set to 1). The first host to reply serves as the ping destination. Type and dimension: string-vector Example: "192.1.0.1"

Table 3-13 Optional attributes for Base mode

Optional attribute	Description
NetworkTimeout	Timeout for ARP and ICMP packets in milliseconds. MultiNICB waits for response to ICMP and ARP packets only during this time period. Assign NetworkTimeout a value in the order of tens of milliseconds (given the ICMP and ARP destinations are required to be on the local network). Increasing this value increases the time for failover. Type and dimension: integer-scalar Default: 100
NoBroadcast	If the value of the attribute is 1, NoBroadcast prevents MultiNICB from sending broadcast ICMP packets. Note that MultiNICB can still send ARP requests. If NetworkHosts are not specified and NoBroadcast is set to 1, the MultiNICB agent cannot function properly. Note: Symantec does not recommend setting the value of NoBroadcast to 1. Type and dimension: integer-scalar Default: 0
OfflineTestRepeatCount	Number of times the test is repeated if the interface status changes from UP to DOWN. For every repetition of the test, the next NetworkHost is selected in round-robin manner. At the end of this process, broadcast is performed if NoBroadcast is set to 0. A greater value prevents spurious changes, but also increases the response time. Type and dimension: integer-scalar Default: 3

Table 3-13 Optional attributes for Base mode

Optional attribute	Description
OnlineTestRepeatCount	Number of times the test is repeated if the interface status changes from DOWN to UP. This test helps to avoid oscillations in the status of the interface. Type and dimension: integer-scalar Default: 3

Optional attributes for Multipathing mode

Table 3-14 Optional attributes for Multipathing mode

Optional attribute	Description
ConfigCheck	If the value of the attribute is 1, the MultiNICB agent checks for: ■ All specified physical interfaces are in the same IP subnet and group, and have "DEPRECATED" and "NOFAILOVER" flags set on them. ■ No other physical interface has the same subnet as the specified interfaces. Valid values for this attribute are 0 and 1. Type and dimension: integer-scalar Default: 1
MpathdRestart	If the value of the attribute is 1, MultiNICB tries to restart mpathd. Valid values for this attribute are 0 and 1. Type and dimension: integer-scalar Default: 1

Resource type definition

```
type MultiNICB (
    static int MonitorInterval = 10
    static int OfflineMonitorInterval = 60
    static str Operations = None
    static str ArgList[] = { UseMpathd, MpathdCommand, ConfigCheck,
        MpathdRestart, Device, NetworkHosts, LinkTestRatio,
        IgnoreLinkStatus, NetworkTimeout, OnlineTestRepeatCount,
        OfflineTestRepeatCount, NoBroadcast, DefaultRouter, Failback,
        GroupName }
    int UseMpathd
    str MpathdCommand = "/sbin/in.mpathd"
    int ConfigCheck = 1
    int MpathdRestart = 1
    str Device{}
    str NetworkHosts[]
    int LinkTestRatio = 1
    int IgnoreLinkStatus = 1
    int NetworkTimeout = 100
    int OnlineTestRepeatCount = 3
    int OfflineTestRepeatCount = 3
    int NoBroadcast
    str DefaultRouter = "0.0.0.0"
    int Failback
    str GroupName
)
```

Solaris operating modes: Base and Multipathing

The MultiNICB agent has two modes of operation, Base and Multipathing, which you can set with the UseMpathd attribute.

Base mode

The value of the UseMpathd attribute is 0 by default for this mode. In Base mode, to monitor the interfaces that it controls, the agent:

- sends the packets to other hosts on the network for probe-based detection
- tests the link status of the interfaces for link-based detection

The agent logs link failures and failovers when it uses either link- or probe-based detection.

If a NIC goes down, the MultiNICB agent notifies the IPMultiNICB agent. The IPMultiNICB agent fails over the virtual IP addresses to a different NIC on the same system. When the original NIC comes up, the agents fail back the virtual IP address.

Each NIC must have its own unique and exclusive base IP address, which the MultiNICB agent uses as the test IP address.

The MultiNICB agent, in Base mode, uses the following criteria to determine if an interface works:

- **Link-based detection of the interface status**
The interface driver reports the status of the link. Note that not all drivers support this feature. Set the value of `IgnoreLinkStatus` to 1 to disable this test.
- **Probe-based detection using Internet Control Message Protocol (ICMP) echo**
Set the `LinkTestRatio` attribute to a value greater than 0 to send ICMP echo request packets to a specified network host. You specify the network hosts in the `NetworkHosts` attribute. You must assign test IP addresses to the interface for probe-based detection. The test IP address is needed to send the ICMP packets, which determines the link's status. If you set the value of the `LinkTestRatio` attribute to 0, you do not need to assign test IP addresses. If you specify no hosts in the `NetworkHosts` attribute, the agent uses the ICMP broadcast when the value of the `NoBroadcast` attribute is 0. It caches the sender of the first reply for future use as a network host. While the agent sends and receives ICMP packets, the IP layer is completely bypassed.

You can assign addresses and still do only link-based detection by setting the values of the `LinkTestRatio` and `IgnoreLinkStatus` attributes to 0. You can skip link-based detection (link driver tests) and only do ICMP tests if:

- the value of the `IgnoreLinkStatus` attribute is 1, and
- the value of the `LimitTestRation` attribute is greater than 0, and
- the test IP addresses are assigned to the interface

The MultiNICB agent performs both link-based detection and probe-based detection if:

- the value of the `LinkTestRatio` attribute is greater than 0, and
- the value of the `IgnoreLinkStatus` attribute is 0, and
- the test IP addresses are assigned to the interface

The MultiNICB agent writes the status of each interface to an export information file, which other agents (like `IPMultiNICB`) or commands (like `haipswitch`) can read.

Failover and fallback

During an interface failure, the MultiNICB agent fails over all logical IP addresses to a working interface under the same resource. The agent remembers the first physical interface from which an IP address was failed over. This physical interface becomes the “original” interface for the particular logical IP address. When the original interface is repaired, the logical IP address fails back to it.

Multipathing mode

To activate this mode set the value of the UseMpathd attribute to 1. The MultiNICB agent, in Multipathing mode, monitors Sun's IP Multipathing daemon (mpathd). The MultiNICB agent specifically monitors the FAILED flag on physical interfaces and the mpathd process. See the man page: `in.mpathd (1M)` for more information on this daemon.

Sun's mpathd daemon monitors the interfaces that are part of the IPMP group. The daemon:

- sends the packets to other hosts on the network for probe-based detection as long as a test IP address is assigned to the network interface
- checks the link status of the interfaces for link-based detection as long as the interface supports the test for detection

The mpathd daemon can perform both link- and probe-based detection when test IP addresses are assigned to NIC interfaces.

The MultiNICB agent logs errors when the daemon is not running, or if a configuration path error exits. The mpathd daemon logs link failures and IP address failovers in the system log.

Trigger script

MultiNICB monitor agent function calls a VCS trigger in case of an interface going up or down. The agent passes the following arguments to the script:

- MultiNICB resource name
- The device whose status changed, for example:
 - qfe0
- The device's previous status (0 for down, 1 for up)
- The device's current status and monitor heartbeat

The agent also sends a notification (which may be received via SNMP or SMTP) to indicate that status of an interface changed. The notification is sent using "health of a cluster resource declined" and "health of a cluster resource improved" traps. These traps are mentioned in the *Veritas Cluster Server User's Guide*. A sample `mnichb_postchange` trigger is provided with the agent. You can customize this sample script as needed or write one from scratch.

The sample script does the following:

- If interface changes status, it prints a message to the console, for example:
MultiNICB: Interface qfe0 came up.
- The script saves last IP address-to-interface name association. If any of the IP addresses have been moved, added, or removed, it prints out a message to the console, for example: MultiNICB: IP address 192.4.3.3 moved from interface qfe1:1 to interface qfe0:1

Sample configurations

Interface configuration

Set the EPROM variable to assign unique MAC addresses to all ethernet interfaces on the host:

```
# eeprom local-mac-address?=true
```

Reboot the system after setting the eeprom variable to complete the address setup. The base IP addresses must be configured on the interfaces before the MultiNICB agent controls the interfaces. You can configure these addresses at system start up using /etc/hostname.XXX initialization files. Refer to the following examples for more information.

Setting up test IP addresses for Base Mode

These examples demonstrate setting up test IP addresses for your clustered systems. These IP addresses allow the agent determine if the NIC works. The agent determines that the NIC works if it receives responses for the ping packets that it sends to other nodes on the network. You do *not* need to perform the following steps for the floating IP addresses. The agent performs these steps.

In the file /etc/hostname.qfe0, add the following two lines:

```
north-qfe0 netmask + broadcast + deprecated -failover up \  
addif north netmask + broadcast + up
```

Where north-qfe0 is the test IP address that the agent uses to determine the state of the qfe0 network card.

In the file /etc/hostname.qfe4, add the following line:

```
north-qfe4 netmask + broadcast + deprecated -failover up
```

Where north-qfe4 is the test IP address that the agent uses to determine the state of the qfe4 network card.

In the example, north-qfe0 and north-qfe4 are the host names that correspond to test IP addresses. north is the host name that corresponds to the test IP address.

IPMultiNICB and MultiNICB configuration

```
cluster clus_north (
    UserNames = { admin = "cDRpdxPmHpzS." }
    Administrators = { admin }
    CounterInterval = 5
)
system north (
)
system south (
)
group g11 (
    SystemList = { north = 0, south = 1 }
    AutoStartList = { north, south }
)
IPMultiNICB g11_i1 (
    BaseResName = gnic_n
    Address = "192.1.0.201"
    NetMask = "255.255.0.0"
    DeviceChoice = "1"
)
Proxy g11_p1 (
    TargetResName = gnic_n
)
g11_i1 requires g11_p1

// A parallel group for the MultiNICB resource

group gnic (
    SystemList = { north = 0, south = 1 }
    AutoStartList = { north, south }
    Parallel = 1
)
MultiNICB gnic_n (
    Device @north = { qfe0, qfe4 }
    Device @south = { qfe0, qfe4 }
    NetworkHosts = { "192.1.0.1" }
)
Phantom gnic_p (
)
```

DNS agent

The DNS agent updates and monitors the mapping for the following:

- The host name to IP address (A, AAAA, or PTR record)
- The canonical name (CNAME)

The agent performs these tasks for a DNS zone when failing over nodes across subnets (a wide-area failover). Resource records (RR) can include different types: A, AAAA, CNAME, NS (name server), SOA, and PTR records.

Use the DNS agent when the failover source and target nodes are on different subnets. The agent updates the name server and allows clients to connect to the failed over instance of the application service.

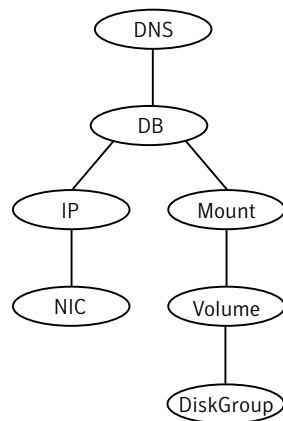
For important information about this agent, refer to:

“[DNS agent notes](#)” on page 105

Dependencies

No dependencies exist for the DNS resource.

Figure 3-7 Sample service group for a DNS resource



Agent functions

Online	Sends a DNS query to retrieve the Start of Authority (SOA) record of the zone that the Domain agent attribute defines. The master server's name is in the SOA field. Unless you define the StealthMasters attribute, it is the only server for the update. When you define the StealthMasters attribute, only the servers that the attribute defines are updated. The agent creates PTR records for each RR of type A or AAAA if the value of the CreatePTR attribute is true. A prerequisite for this feature is that the same master or stealth servers serve the forward (A or AAAA) and reverse zones.
Offline	Removes the Online lock file. If attribute OffDelRR is true, offline removes all records that the ResRecord keys define.
Monitor	Returns the ONLINE state if at least one name server reports all mappings that ResRecord or Hostname and Alias defines. The name servers are the master or StealthMaster, and all the servers for which an NS record for the zone exists.
Clean	Removes the Online lock file, if it exists.
Open	Removes the Online lock file if the resource is reported online on another node inside the cluster to prevent concurrency violation. If the lock file exists, at least one name server has to report all the RRs that the ResRecord or Hostname and Alias attributes define. If one name server cannot report all the RRs, the agent function removes the Online lock file.
Action	Different action agent functions follow: ■ keyfile.vfd This action entry point checks if the key file as specified in the TSIGKeyFile attribute exists either locally or on shared storage. ■ dig.vfd This action entry point checks if dig and nsupdate binaries exist and are executable. ■ master.vfd This action entry point checks if stealth masters are pingable from the node.

State definitions

ONLINE	Online lock file exists and servers returning all configured resource records.
OFFLINE	Indicates an offline state when either of the following is true: ■ The online lock does not exist. ■ At least one server cannot report all of the RRs' mappings.
UNKNOWN	A problem exists with the configuration. Can indicate that the resource record list contains an invalid value as a part of the record key or a record value of the ResRecord attribute.

Attributes

Table 3-15 Required attributes

Required attribute	Description
Domain	A string representing the DNS zone that the agent administers. The domain name can only contain alphanumeric symbols and the dash. Type and dimension: string-scalar Examples: ■ Forward mapping "demo.symantec.com" ■ IPv4 reverse mapping "2.168.192.in-addr.arpa"
■ Hostname and Alias or ■ ResRecord	You must use either the ResRecord attribute only or the HostName and Alias attributes. Do not use all three attributes together.
Alias	A string representing the alias to the canonical name. Type and dimension: string-scalar Example: "www" Where www is the alias to the canonical name mtv.symantec.com. See "Sample Web server configuration" on page 106.
Hostname	A string that represents the canonical name of a system. Type and dimension: string-scalar Example: "mtv.symantec.com"

Table 3-15 Required attributes

Required attribute	Description
ResRecord	You can use the ResRecord attribute alone, or you can use the Hostname and Alias attributes. ResRecord is an association of DNS resource record values. Each ResRecord attribute consists of two values: <i>DNS record key</i> = <i>DNS record data</i>. Note that the record key must be a unique value. If the resource record list contains any invalid value as a part of the record key or a record value of the ResRecord attribute, the resource enters an UNKNOWN state. Type and dimension: association-scalar Examples: ■ For forward mapping, where the zone is demo.symantec.com: - sles901 = "192.168.2.191" - ww2 = sles901 - sles9ip6 = "2007::1:2:3:abc" ■ A multi-home DNS record, typically for one host with two network interfaces, different address, but the same DNS name. This results in two-A records, or a single A record with continuation lines. sle902 = "192.168.2.102 10.87.13.22" A multi-home AAAA DNS record can be configured as below: sle902 = "1234::5678 1234::AABB:CCDD" ■ For reverse IPv4 address mapping, where the zone is 2.168.192.in-addr.arpa: 191 = "sles901.demo.symantec.com" ■ For reverse IPv6 address mapping, where the zone is 3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.7.0.0.2.ip6.arpa: cba = "sles9ip6.demo.symantec.com" Use only partial host names. If you use a fully qualified domain name, append a period "." at the end of the name. For CNAME records, use: ■ ResRecord = { www = mydesktop } - or ■ ResRecord = { www = "mydesktop.marketing.db.com." } Where the Domain attribute is "marketing.db.com"

Table 3-16 Required attributes

Required attribute	Description
ResRecord (cont.)	The agent uses case-insensitive pattern matching—and a combination of the Domain and ResRecord attribute values—to determine the resource record type. The RR type is as follows: ■ PTR: if the Domain attribute ends with .arpa ■ A: if the record data field is four sets of numbers, where a space separates each set. The following details the pattern it tries to match: [1-223].[0-255].[0-255].[0-255] Hexadecimal is not supported. ■ AAAA: if the record data fields are in multiple sets of hexadecimal format, then this record is an IPv6 associated type AAAA record. ■ CNAME: for any other valid record data. Note: If a name in the ResRecord attribute does not comply with RFC 1035, then a warning is issued to the log file. The ResRecord association is not used.

Table 3-17 Optional attributes

Optional attribute	Description
TTL	A non-zero integer represents the “Time To Live” value, in seconds, for the DNS entries in the zone that you want to update. A lower value means more hits on your DNS server, while a higher value means more time for your clients to learn about changes. The time-in-seconds value may take the value 0, which indicates never caching the record, to a maximum of 2,147,483,647, which is over 68 years! The current best practice recommendation (RFC 1912) proposes a value greater than one day, and on RRs that do not change often, consider multi-week values. Type and dimension: integer-scalar Default: 86400 Example: "3600"

Table 3-17 Optional attributes

Optional attribute	Description
StealthMasters	The list of primary master name servers in the domain. This attribute is optional since the first name server is retrieved from the zone's SOA (Start of Authority) record. If the primary master name server is a stealth server, define this attribute. A stealth server is a name server that is authoritative for a zone, but does not appear in that zone's SOA record. It is hidden to prevent direct attacks from the Internet. Type and dimension: string-keylist Example: { "10.190.112.23" }
TSIGKeyFile	Required when you configure DNS for secure updates. Specifies the absolute path to the file containing the private TSIG (Transaction Signature) key. Type and dimension: string-scalar Example: /var/tsig/example.com.+157+00000.private
CreatePTR	Use the CreatePTR attribute to direct the online agent function to create PTR records for each RR of type A or AAAA. You must set the value of this attribute to true (1) to create the records. Before you can use this attribute, the same master or stealth servers must serve the forward (A or AAAA) and reverse zones. Type and dimension: boolean-scalar Default: 0 Example: 1
OffDelRR	Use the OffDelRR attribute to direct the offline agent function to remove all records that the ResRecord key defines. You must set the value of this attribute to true (1) to have the agent remove all the records. The online agent function always adds records if they do not exist. Type and dimension: boolean-scalar Default: 0 Example: 1

Resource type definition

```
type DNS (  
    static keylist SupportedActions = { "dig.vfd", "keyfile.vfd",  
        "master.vfd" }  
    static str ArgList[] = { Domain, Alias, Hostname, TTL,  
        TSIGKeyFile, StealthMasters, ResRecord, CreatePTR, OffDelRR }  
    str Domain  
    str Alias  
    str Hostname  
    int TTL = 86400  
    str TSIGKeyFile  
    str StealthMasters[]  
    str ResRecord{}  
    boolean CreatePTR = 0  
    boolean OffDelRR = 0  
)
```

DNS agent notes

The DNS agent has the following notes:

- [“High availability fire drill”](#) on page 105
- [“Monitor scenarios”](#) on page 106
- [“Sample Web server configuration”](#) on page 106
- [“Secure DNS update for BIND 9”](#) on page 106
- [“Setting up secure updates using TSIG keys for BIND 9”](#) on page 107

High availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node; discrepancies that might prevent a service group from going online on a specific node.

For DNS resources, the high availability drill tests the following conditions:

- Checks if the key file as specified by the TSIGKeyFile attribute is available either locally or on shared storage.
- Checks if the dig and nsupdate binaries are available on the cluster node and are executable on that node.
- Checks if the stealth masters are pingable from the cluster node so as to ensure that there is no network issue that would prohibit the DNS update and query requests from reaching the stealth master server.

For more information about using the high availability fire drill see the *Veritas Cluster Server User’s Guide*.

Monitor scenarios

Depending on the existence of the Online lock file and the defined Resource Records (RR), you get different status messages from the Monitor function.

Table 3-18 Monitor scenarios for the Online lock file

Online lock file exists	Expected RR mapping	Monitor returns
NO	N/A	OFFLINE
YES	NO	OFFLINE
YES	YES	ONLINE

Sample Web server configuration

Take the former Veritas corporate web server as an example. A browser requests the URL `http://www.veritas.com` that maps to the canonical name `mtv.veritas.com`. The browser retrieves the IP address for the web server by querying a domain name server. If the web server fails over from Mountain View to Heathrow (`hro.veritas.com`), the domain name servers need a new canonical name mapping for `www.veritas.com`. The `www.veritas.com` alias is now updated to point to the canonical name of the standby system in Heathrow.

Secure DNS update for BIND 9

The DNS agent expects that the zone’s allow-update field contains the IP address for the hosts that can dynamically update the DNS records. This functionality is default for the DNS agent. Since a competent black hat can, however, spoof IP addresses, consider TSIG as an alternative.

TSIG (Transaction Signature) as specified in RFC 2845 is a shared key message authentication mechanism that is available in DNS. A TSIG key provides the means to authenticate and verify the validity of exchanged DNS data. It uses a shared secret key between a resolver and either one or two servers to provide security.

Setting up secure updates using TSIG keys for BIND 9

In the following example, the domain is example.com.

To use secure updates using TSIG keys

- 1 Run the `dnssec-keygen` command with the HMAC-MD5 option to generate a pair of files that contain the TSIG key:

```
# dnssec-keygen -a HMAC-MD5 -n HOST example.com.  
example.com.+157+00000
```

- 2 Open the `example.com.+157+00000.key` file. After you run the `cat` command, the contents of the file resembles:

```
# cat example.com.+157+00000.key  
example.com. IN KEY 512 3 157 +Cdjlkef9ZTSeixERZ433Q==
```

- 3 Copy the shared secret (the TSIG key), which looks like:

```
+Cdjlkef9ZTSeixERZ433Q==
```

- 4 Configure the DNS server to only allow TSIG updates using the generated key. Open the `named.conf` file and add these lines.

```
key example.com. {  
    algorithm hmac-md5;  
    secret "+Cdjlkef9ZTSeixERZ433Q==";  
};
```

Where `+Cdjlkef9ZTSeixERZ433Q==` is the key.

- 5 In the `named.conf` file, edit the appropriate zone section and add the `allow-update` sub-statement to reference the key:

```
allow-update { key example.com. ; } ;
```

- 6 Save and restart the `named` process.

- 7 Place the files containing the keys on each of the nodes that is listed in your group's `SystemList`. The DNS agent uses this key to update the name server. Copy both the private and public key files on to the node. A good location is in the `/var/tsig/` directory.

- 8 Set the `TSIGKeyFile` attribute for the DNS resource to specify the file containing the private key.

```
DNS www (  
    Domain = "example.com"  
    ResRecord = {www = north}  
    TSIGKeyFile = "/var/tsig/example.com.+157+00000.private"  
)
```


File share agents

This chapter contains the following:

- [“About the file service agents”](#) on page 109
- [“NFS agent”](#) on page 110
- [“NFSRestart agent”](#) on page 115
- [“Share agent”](#) on page 123
- [“About the Samba agents”](#) on page 126
- [“SambaServer agent”](#) on page 128
- [“SambaShare agent”](#) on page 131
- [“NetBIOS agent”](#) on page 134

About the file service agents

Use the file service agents to provide high availability for file share resources.

NFS agent

Starts and monitors the nfsd and mountd daemons required by all exported NFS file systems.

Symantec recommends that you configure only one NFS resource in a service group on a node. If you have more than one service group that uses the NFS resource, the other service groups can use a Proxy resource. The Proxy resource can point to the NFS resource in the first group. This use of the Proxy resource prevents redundant monitoring of the NFS daemons on the same system.

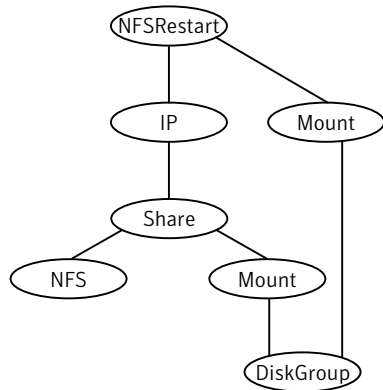
For important information about this agent, refer to:

[“NFS agent notes”](#) on page 113

Dependencies

The NFS resource does not depend on other resources.

Figure 4-1 Sample service group for an NFS resource



Agent functions

Online	Checks if nfsd, mountd, and nfsmapid (nfsmapid is for Solaris 10) daemons are running. If they are not running, the agent starts the daemons.
Monitor	Monitors versions 2, 3, and 4 of the nfsd daemons, and versions 1, 2, and 3 of the mountd daemons. Monitors TCP and UDP versions of the daemons by sending RPC (Remote Procedure Call) calls <code>clnt_create</code> and <code>clnt_call</code> to the RPC server. If the calls succeed, the resource is reported ONLINE.
Clean	Terminates and restarts the nfsd, mountd, and nfsmapid daemons.

State definitions

ONLINE	Indicates that the NFS daemons are running in accordance with the supported protocols and versions.
OFFLINE	Indicates that the NFS daemons are not running in accordance with the supported protocols and versions.
FAULTED	Indicates that the NFS daemons are not running in accordance with the supported protocols and versions.
UNKNOWN	Unable to determine the status of the NFS daemons.

Attributes

Optional attributes

Optional attributes	Description
LockFileTimeout	Specifies the time period in seconds after which the agent deletes the lock files. The agent maintains the files internally to synchronize the starting and stopping of NFS daemons between multiple service groups. Set this value to the total time needed for a service group to go offline or come online on a node. In situations where you have multiple service groups, set this value for the service group that takes the longest time. Type and dimension: integer-scalar Default: 180 Example: "240"
Nservers	Specifies the number of concurrent NFS requests the server can handle. Type and dimension: integer-scalar Default: 16 Example: "24"

Resource type definition

```
type NFS (  
  static int RestartLimit = 1  
  static str ArgList[] = { Nservers, LockFileTimeout }  
  static str Operations = OnOnly  
  int Nservers = 16  
  int LockFileTimeout = 180  
)
```

NFS agent notes

The NFS agent has the following notes:

- [“Service Management Facility for Solaris 10”](#) on page 113

Service Management Facility for Solaris 10

You must disable the Service Management Facility (SMF) for NFS daemons for the NFS agent to work on Solaris 10. SMF is the new service framework for Solaris 10. SMF provides an infrastructure to automatically start and restart services.

UNIX start-up scripts and configuration files previously performed these functions. SMF maintains the Service Configuration Repository to store persistent configuration information as well as runtime data for all the services. All NFS daemons (nfsd, mountd, etc.) are now controlled by SMF. To keep these daemons under VCS control, modify the configuration repository to disable the SMF framework for NFS daemons.

You must invoke the following command before bringing the NFS agent online or the agents returns an UNKNOWN state.

To keep nfsd and mountd daemons under VCS control

- 1 Set the auto_enable property to false.

```
# svccfg -s nfs/server setprop "application/auto_enable = false"
```
- 2 Refresh the SMF configuration repository.

```
# svcadm refresh nfs/server
```
- 3 Disable SMF.

```
# svcadm disable svc:/network/nfs/server:default
```
- 4 Run `svcs -a | grep -i nfs` command and review its output to make sure that SMF for nfsd and mountd is disabled.

```
# svcs -a | grep -i nfs
disabled          May_29    svc:/network/nfs/server:default
```

To keep nfsmapid daemon under VCS control

- 1 Set the auto_enable property to false.
svccfg -s nfs/mapid setprop "application/auto_enable = false"
- 2 Refresh the SMF configuration repository.
svcadm refresh nfs/mapid
- 3 Disable SMF.
svcadm disable svc:/network/nfs/mapid:default
- 4 Run **svcs -a | grep -i mapid** command and review its output to make sure that SMF for nfsmapid is disabled.
svcs -a | grep -i mapid
disabled May_29 svc:/network/nfs/mapid:default

Sample configurations

Configuration

```
NFS NFS_groupx_24 (
  Nservers = 24
  LockFileTimeout = 240
)
```

NFSRestart agent

The NFSRestart agent recovers NFS record locks after sudden reboots or crashes on clients and servers. This avoids file corruption and provides high availability for NFS record locks.

The NFSRestart agent brings online, takes offline, and monitors the three daemons: `smSyncd`, `statd`, and `lockd`.

If you have configured the NFSRestart agent for lock recovery, the NFSRestart agent starts the `smSyncd` daemon. The daemon copies the NFS locks from the shared-storage to the local directory (`/var/statmon/sm`) and vice-versa.

The NFSRestart agent brings online, takes offline, and monitors the three daemons: `smSyncd`, `statd`, and `lockd`.

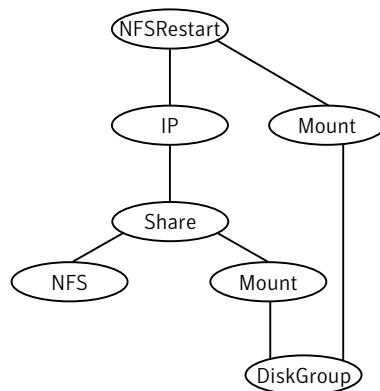
For important information about this agent, refer to:

[“NFSRestart agent notes”](#) on page 118

Dependencies

This resource must be at the top of the resource dependency tree of a service group. Only one NFSRestart resource should be configured in a service group. The NFSRestart, NFS, and Share agents must be in same service group.

Figure 4-2 Sample service group for an NFSRestart resource



Agent functions

Online	■ Terminates statd and lockd. ■ If the value of the NFSLockFailover attribute is 1, it copies the locks from the shared storage to the /var/statmon/sm directory for Solaris. ■ Copies the locks from the shared storage to the /var/statmon/sm directory if NFSLockFailover is set to 1. ■ Starts the statd and lockd daemons. ■ Starts the smsyncd daemon to copy the contents of the /var/statmon/sm directory to the shared storage (LocksPathName) at regular, two-second intervals if the value of the NFSLockFailover attribute is 1.
Monitor	Monitors the statd and lockd daemons and restarts them if they are not running. It also monitors the smsyncd daemon if the value of the NFSLockFailover attribute is 1.
Offline	■ Terminates the statd and lockd daemons to clear the lock state. ■ Terminates the nfsd and mountd daemons to close the TCP/IP connections. ■ Terminates the smsyncd daemon if the daemon is running.
Clean	■ Terminates the statd and lockd daemons to clear the lock state. ■ Terminates the nfsd and mountd daemons to close TCP/IP connections. ■ Terminates the smsyncd daemon if the daemon is running.
Action	■ nfsconf.vfd Checks the NFS configuration file to confirm that the NFS server does not come online automatically after reboot. ■ lockdir.vfd Verifies that the NFS lock directory (which is specified by the LocksPathName attribute of NFSRestart) is on shared storage.
nfs_postoffline	Restarts nfsd, mountd, lockd, statd, and nfsmapid after the group goes offline.

State definitions

ONLINE	Indicates that the daemons are running properly.
OFFLINE	Indicates that one or more daemons are not running.
UNKNOWN	Indicates the inability to determine the agent's status.

Attributes

Table 4-1 Optional attributes

Required attribute	Description
LocksPathName	The path name of the directory to store the NFSLocks for all the shared filesystems. You can use the pathname of one of the shared file systems for this value. Type and dimension: string-scalar Example: "/share1x"
NFSLockFailover	A flag that specifies whether the user wants NFS Locks to be recovered after a failover Type and dimension: boolean-scalar Default: 0

Table 4-2 Required attributes

Required attribute	Description
NFSRes	Name of the NFS resource. Do not set this to the name of the Proxy resource that points to the NFS resource. Type and dimension: string-scalar Example: "nfsres1"

Resource type definition

```
type NFSRestart (
  static keylist SupportedActions = { "lockdir.vfd",
    "nfsconf.vfd" }
  static str ArgList[] = { LocksPathName, NFSLockFailover, NFSRes,
    "NFSRes:LockFileTimeout" }
  str NFSRes
  str LocksPathName
  boolean NFSLockFailover = 0
)
```

NFSRestart agent notes

The NFSRestart agent has the following notes:

- [“High availability fire drill”](#) on page 118
- [“Service Management Facility—Solaris 10”](#) on page 118
- [“Providing a fully qualified host name”](#) on page 120

High availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node; discrepancies that might prevent a service group from going online on a specific node. For NFSRestart resources, the high availability drill performs the following, it:

- Checks the NFS configuration file to confirm that the NFS server does not come online automatically after reboot.
- Verifies that the NFS lock directory (which is specified by the `LocksPathName` attribute of NFSRestart) is on shared storage.

For more information about using the high availability fire drill see the *Veritas Cluster Server User’s Guide*.

Service Management Facility—Solaris 10

You must disable the Service Management Facility (SMF) for NFS daemons for the NFSRestart agent to work on Solaris 10. SMF is the new service framework for Solaris 10 starting from build 64. SMF provides an infrastructure to automatically start and restart services. Previously, UNIX start-up scripts and configuration files performed these functions.

SMF maintains the Service Configuration Repository, which stores persistent configuration information and runtime data for all the services. Thus, SMF now controls all NFS locking daemons (`lockd`, `statd`, etc.) To keep these daemons under VCS control, you need to modify the configuration repository to disable the SMF framework for NFS daemons.

You must invoke the following command before bringing the NFSRestart agent online or the agents returns an UNKNOWN state.

To keep the statd daemon under VCS control

- 1 Set the auto_enable property to false.

```
# svccfg -s nfs/status setprop "application/auto_enable = false"
```
- 2 Refresh the SMF configuration repository.

```
# svcadm refresh nfs/status
```
- 3 Disable SMF.

```
# svcadm disable svc:/network/nfs/status:default
```
- 4 Run `svcs -a | grep -i nfs` command and review its output to make sure that SMF for statd is disabled.

```
# svcs -a | grep -i nfs
disabled      May_29    svc:/network/nfs/status:default
```

To keep lockd daemon under VCS control

- 1 Set the auto_enable property to false.

```
# svccfg -s nfs/nlockmgr setprop "application/auto_enable = false"
```
- 2 Refresh the SMF configuration repository.

```
# svcadm refresh nfs/nlockmgr
```
- 3 Disable SMF.

```
# svcadm disable svc:/network/nfs/nlockmgr:default
```
- 4 Run `svcs -a | grep -i nfs` command and review its output to make sure that SMF for nlockmgr is disabled.

```
# svcs -a | grep -i nlockmgr
disabled      May_29    svc:/network/nfs/nlockmgr:default
```

To manually restart lockd, statd, and automountd

- For lockd:

```
# /usr/lib/nfs/lockd
```
- For statd:

```
# /usr/lib/nfs/statd
```
- For automountd:

```
# /usr/lib/fs/autofs/automount
# /usr/lib/autofs/automountd
```

Providing a fully qualified host name

You must provide a fully qualified host name (nfsserver.princeton.edu) for the NFS server while mounting the file system on the NFS client. If you do not use a fully qualified host name, or if you use a virtual IP address (10.122.12.25) or partial host name (nfsserver), NFS lock recovery fails.

If you want to use the virtual IP address or a partial host name, make the following changes to the service database (hosts) and the nsswitch.conf files:

```
/etc/hosts
```

To use the virtual IP address and partial host name for the NFS server, you need to add an entry to the /etc/hosts file. The virtual IP address and the partial host name should resolve to the fully qualified host name.

```
/etc/nsswitch.conf
```

You should also modify the hosts entry in this file so that upon resolving a name locally, the host does not first contact NIS/DNS, but instead immediately returns a successful status. Changing the nsswitch.conf file might affect other services running on the system.

For example:

```
hosts: files [SUCCESS=return] dns nis
```

You have to make sure that the NFS client stores the same information for the NFS server as the client uses while mounting the file system. For example, if the NFS client mounts the file system using fully qualified domain names for the NFS server, then the NFS client directory: /var/statmon/sm directory should also have a fully qualified domain name after the acquisition of locks.

Otherwise, you need to start and stop the NFS client twice using the /etc/init.d/nfs.client script to clear the lock cache of the NFS client.

A time period exists where the virtual IP address is online but locking services are not registered on the server. Any NFS client trying to acquire a lock in this interval would fail and get ENOLCK error.

Every two seconds, the smsyncd daemon copies the list of clients that hold the locks on the shared filesystem in the service group. If the service group fails before smsyncd has a chance to copy the client list, the clients may not get a notification once the service group is brought up. This causes NFS lock recovery failure.

Sample configurations

```
include "types.cf"

cluster nfscclus (
    UserNames = { admin = joe }
    Administrators = { admin }
)

system sysA (
)

system sysB (
)

group nfsres_grp (
    SystemList = { sysA = 0, sysB = 1 }
)

DiskGroup dg (
    DiskGroup = nfsrc_dg
)

IP ip (
    Device = bge0
    Address = "11.152.6.155"
    NetMask = "255.255.240.0"
)

Mount mnt (
    MountPoint = "/nfsrc_mnt"
    BlockDevice = "/dev/vx/dsk/nfsrc_dg/nfsrc_vol"
    FSType = vxfs
    MountOpt = rw
    FsyncOpt = "-y"
)

NFS nfs (
)

NFSRestart nfsres (
    LocksPathName = "/nfsrc_mnt"
    NFSLockFailover = 1
    NFSRes = "nfs"
)

Share share (
    PathName = "/nfsrc_mnt"
    Options = "-o rw"
)
```

```
Volume vol (
    Volume = nfsr_vol
    DiskGroup = nfsr_dg
)

ip requires share
mnt requires vol
nfsres requires ip
share requires mnt
share requires nfs
vol requires dg

// resource dependency tree
//
//     group nfsres_grp
//     {
//         NFSRestart nfsres
//         {
//             IP ip
//             {
//                 Share share
//                 {
//                     Mount mnt
//                     {
//                         Volume vol
//                         {
//                             DiskGroup dg
//                         }
//                     }
//                 }
//             }
//             NFS nfs
//         }
//     }
// }
```

Share agent

Shares, unshares, and monitors a single local resource for exporting an NFS file system to be mounted by remote systems.

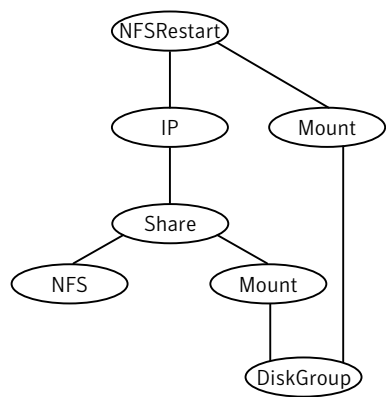
Before you use this agent, verify that the files and directories to be shared are on shared disks.

For important information on this agent, refer to:
“[Share agent notes](#)” on page 125

Dependencies

Share resources depend on NFS. In NFS service group, IP, IPMultiNIC, and IPMultiNICB resources depend on Share resources.

Figure 4-3 Sample service group for a Share resource



Agent functions

Online	Shares an NFS file system.
Offline	Unshares an NFS file system.
Monitor	Reads /etc/dfs/sharetab file and looks for an entry for the file system specified by PathName. If the entry exists, monitor returns ONLINE.
Action	direxists.vfd Checks if the path specified by the PathName attribute exists on the cluster node. If the path name is not specified, it checks if a corresponding mount point is available to ensure that the path is on shared storage.

State definitions

ONLINE	Indicates that specified directory is exported to the client.
OFFLINE	Indicates that the specified directory is not exported to the client.
UNKNOWN	Indicates that the agent could not determine the state of the resource or that the resource attributes are invalid.
FAULTED	Indicates that the share has unexported outside of VCS control.

Attributes

Table 4-3 Required attributes

Required attribute	Description
PathName	Pathname of the file system to be shared. Type and dimension: string-scalar Example: "/share1x"

Table 4-4 Optional attributes

Optional attribute	Description
Options	Options for the <code>share</code> command. Type and dimension: string-scalar Example: "-o rw"

Resource type definition

```
type Share (  
    static keylist SupportedActions = { "direxists.vfd" }  
    static str ArgList[] = { PathName, Options }  
    str PathName  
    str Options  
)
```

Share agent notes

The following section contains notes on the Share agent.

High availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node. These discrepancies might prevent a service group from going online on a specific node. For Share resources, the high availability fire drill checks if the path exists.

For more information about using the high availability fire drill see the *Veritas Cluster Server User's Guide*.

Sample configurations

Configuration

```
Share nfsshare1x (  
    PathName = "/share1x"  
)
```

About the Samba agents

Samba is a suite of programs that allows a system running a UNIX or UNIX-like operating system to provide services using the Microsoft network protocol. Samba supports the following services:

- Filespace
- Printer
- WINS
- Domain Master

Configure these services in the Samba configuration file (`smb.conf`). Samba uses two processes: `smbd` and `nmbd` to provide these services.

VCS provides Samba failover using three agents: `SambaServer`, `NetBios`, and `SambaShare`.

The Samba agents

- The `NetBios` agent
- The `SambaServer` agent
- The `SambaShare` agent

Before using the Samba agents

- Verify that `smbd` and `nmbd` always run as daemons. Verify that they cannot be started using the meta-daemon `inetd`.
- Verify that the `smbd` and `nmbd` daemons are in the `path` environment variable.
- If they are not, verify that they run from the default directory `/usr/bin`.
 - The path of `smbd` and `nmbd` is `/usr/sfw/sbin`.
- Verify that Samba is configured properly and that the Samba configuration file is identical on all cluster systems. The user can replicate the file or store it on a shared disk accessible from all cluster systems.
- If configuring Samba as a WINS server or Domain Master, verify that the Samba lock directory is on the shared disk. This ensures that the WINS server database and Domain Master are created on the shared disk.

Supported versions

Table 4-5 provides the support matrix for the Samba agents.

Table 4-5 Supported platforms, architectures, and Samba versions

Platforms	Operating systems/ Architecture	Supported Samba versions
Solaris	Solaris 8, 9, and 10 (32-bit and 64-bit)/SPARC	Supports the version that is bundled with the operating system.
	Solaris 10 (64-bit)/x64	n/a

Configuring the Samba agents

If Samba is configured properly, and the configuration file is identical on all cluster systems, configure resources of type SambaServer and NetBios only. This ensures that all shares in the Samba configuration file are failed over when the SambaServer resource fails over. Note that the Samba shares are not monitored. To monitor the Samba shares, configure the agents with the following dependencies:

```
SambaShare requires NetBios
SambaShare requires SambaServer
NetBios requies IP
```

For example, use the following configuration to monitor Samba shares SambaShare1 and SambaShare2. Use multiple resources of type SambaShare (if necessary), but only one resource each of type NetBios and SambaServer.

```
SambaShare1 requires NetBios1
SambaShare1 requires SambaServer1
SambaShare2 requires NetBios1
SambaShare2 requires SambaServer1
NetBios1 requies IP_1
```

SambaServer agent

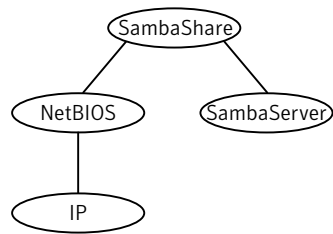
The SmbServer agent starts, stops, and monitors the smbd process as a daemon. Only one resource of this type is permitted. You can use the agent to make a smbd daemon highly available or to monitor it.

The smbd daemon provides Samba share services. The agent makes a copy of smbd for each client and verifies that Samba is running by reading the pid of this daemon. The agent can perform in-depth monitoring by establishing a socket connection to Samba at ports where the daemon is listening and sending it a NetBIOS session request.

Dependencies

No dependencies exist for the SambaServer resource.

Figure 4-4 Sample service group for a SambaServer resource



Agent functions

Online	Starts the smbd daemon at specified ports.
Offline	Stops the smbd daemon.
Monitor	Verifies that the smbd daemon is running by reading its pid file. Does in-depth monitoring periodically, if configured, by establishing a socket connection to Samba and sending it a NetBIOS session request.
Clean	Stops the smbd daemon.

State definitions

ONLINE	Indicates that the smbd daemon is running. If in-depth monitoring is configured, it indicates that a positive session response packet was received through a socket connection to the Samba server.
OFFLINE	Indicates that smbd is not running. If in-depth monitoring is enabled, it indicates that the agent could not establish a socket connection with the server, or that it received an incorrect response packet header, or the session response packet connection timed out.
UNKNOWN	Indicates that the agent could not determine the state of the resource.

Attributes

Table 4-6 Required attributes

Required attribute	Description
ConfFile	Complete path of the configuration file that Samba uses. Type and dimension: string-scalar Example: "/etc/sfw/smb.conf"
LockDir	Lock directory of Samba. Samba stores the files smbd.pid, nmbd.pid, wins.dat (WINS database), and browse.dat (master browser database) in this directory. Type and dimension: string-scalar Example: "/var/samba/locks"
SambaTopDir	Parent path of Samba daemon and binaries. Example: "/usr/sfw"

Resource type definitions

```
type SambaServer (  
    static int RestartLimit = 5  
    static str ArgList[] = { ConfFile, SambaTopDir, LockDir, Ports,  
        IndepthMonitorCyclePeriod, ResponseTimeout }  
    str ConfFile = "/etc/sfw/smb.conf"  
    str SambaTopDir = "/usr/sfw"  
    str LockDir = "/var/samba/locks"  
    int Ports[] = { 139, 445 }  
    int IndepthMonitorCyclePeriod = 5  
    int ResponseTimeout = 10  
)
```

Sample configurations

```
SambaServer samba_server (  
    ConfFile = "/etc/sfw/smb.conf"  
    LockDir = "/var/samba/locks"  
    SambaTopDir = "/usr/sfw"  
    IndepthMonitorCyclePeriod = 3  
    ResponseTimeout = 15  
)
```

SambaShare agent

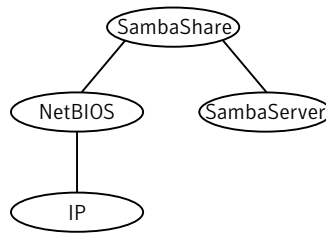
The SambaShare agent adds, removes, and monitors a share by modifying the specified Samba configuration file. You can use the agent to make a Samba Share highly available or to monitor it.

Each filesystem or printer service provided by Samba is a shared resource and is defined as a section in the Samba configuration file. The section name is the name of the shared resource and the section parameters define the share attributes.

Dependencies

SambaShare resources depend on SambaServer, NetBios and Mount resources.

Figure 4-5 Sample service group for a SambaShare resource



Agent functions

Online	Edits the samba configuration file and adds the shares.
Offline	Removes the shares from the configuration file.
Monitor	Issues the command <code>smbclient</code> to check if the specified shares exist.
Clean	Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

State definitions

ONLINE	Indicates that the share is available and that the share path exists.
OFFLINE	Indicates that the share is not available, or that the share has a non-existent path.
UNKNOWN	Indicates that the agent could not determine the state of the resource.

Attributes

Table 4-7 Required attributes

Required attribute	Description
SambaServerRes	Name of the SambaServer resource. Type and dimension: string-scalar Example: "SG.smb_res1" Where SG is the service group to which the resource smb_res1 belongs.
"SambaServerRes:ConfFile"	Complete path of the configuration file that is specified in the SambaServer resource.
"SambaServerRes:LockDir"	Complete path of the lock directory that is specified in the SambaServer resource.
"SambaServerRes:SambaTopDir"	Parent path of Samba daemon and binaries installed.
ShareName	Name of the share resource. Type and dimension: string-scalar Example: "share1"
ShareOptions	List of parameters for the share attributes. These parameters are specified as name=value pairs, with each pair separated by a semicolon (;). Type and dimension: string-scalar Example: "path=/shared; public=yes; writable=yes"

Resource type definition

```
type SambaShare (  
  static str ArgList[] = { "SambaServerRes:ConfFile",  
    "SambaServerRes:SambaTopDir", "SambaServerRes:LockDir",  
    ShareName, ShareOptions, "SambaServerRes:Ports" }  
  str SambaServerRes  
  str ShareName  
  str ShareOptions  
)
```

Sample configuration

```
SambaShare Samba_SambaShare3 (  
  SambaServerRes = Samba_SambaServer  
  ShareName = smbshare3  
  ShareOptions = "path=/smbshare3; public=yes; writable=yes"  
)
```

NetBIOS agent

The NetBIOS agent starts, stops, and monitors the `nmbd` daemon. Only one resource of this type is permitted. You can use the agent to make the `nmbd` daemon highly available or to monitor it.

The agent sets, monitors, and resets the names and network interfaces by which the Samba server is known. The agent also sets, monitors and resets Samba to act as a WINS server or domain master or both.

Note that `nmbd` broadcasts the NetBIOS name, or the name by which the Samba server is known in the network.

Before using this agent:

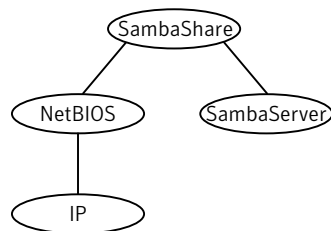
- Set the NetBIOS name.
- Set the NetBIOS interface.

Dependencies

The NetBios resource depends on the IP, the IPMultiNIC or the IPMultiNICB resource.

Note: You can configure only one NetBios resource on a system.

Figure 4-6 Sample service group for a NetBIOS resource



Agent functions

Online	Updates the Samba configuration with the NetBIOS name, all NetBIOS aliases and network interfaces, WINS support, and domain master options specified in the NetBIOS resource. Starts the nmbd daemon.
Offline	Removes the NetBIOS name, all NetBIOS aliases and network interfaces, WINS support, and domain master options specified in the NetBIOS resource from the Samba configuration file. Stops the nmbd daemon.
Monitor	Verifies that the Samba configuration contains the NetBIOS name, all NetBIOS aliases and network interfaces, WINS support, and domain master options specified in the NetBIOS resource.
Clean	Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

State definitions

ONLINE	Indicates that the specified NetBIOS aliases are advertised and that Samba is handling requests for all specified network interfaces. Indicates that WINS and Domain support services are running, if configured.
OFFLINE	Indicates one or more of the following: ■ NetBIOS name is not advertised. ■ A NetBIOS alias is not advertised. ■ Samba is not handling requests on one of the specified interfaces. ■ If WINS support is configured, Samba is not providing WINS service. ■ If domain support is set, Samba is not providing Domain Master service.
UNKNOWN	Indicates that the agent could not determine the state of the resource.

Attributes

Table 4-8 Required attributes

Required attribute	Description
NetBiosName	Name by which the Samba server is known in the network. Type and dimension: string-scalar
"SambaServerRes:ConfFile"	Complete path of the configuration file that is specified in the SambaServer resource. Type and dimension: string-scalar
"SambaServerRes:LockDir"	Complete path of the lock directory that is specified in the SambaServer resource.
SambaServerRes:SambaTopDir	Parent path of Samba daemon and binaries installed.

Table 4-9 Optional attributes

Optional attribute	Description
Interfaces	List of network interfaces on which Samba handles browsing. Type and dimension: string-vector Example: "172.29.9.24/16"
NetBiosAliases	List of additional names by which the Samba server is known in the network. Type and dimension: string-vector Example: "host1_samba, myname"

Table 4-9 Optional attributes

Optional attribute	Description
WinsSupport	If set to 1, this flag causes the agent to configure Samba as a WINS server. Type and dimension: integer-scalar Default: 0

Resource type definition

```
type NetBios (  
  static str ArgList[] = { "SambaServerRes:ConfFile",  
    "SambaServerRes:SambaTopDir", "SambaServerRes:LockDir",  
    NetBiosName, NetBiosAliases, Interfaces, WinsSupport,  
    DomainMaster }  
  str SambaServerRes  
  str NetBiosName  
  str NetBiosAliases[]  
  str Interfaces[]  
  int WinsSupport  
  int DomainMaster  
)
```

Sample configuration

```
NetBios Samba_NetBios (  
  SambaServerRes = Samba_SambaServer  
  NetBiosName = samba_demon  
  NetBiosAliases = { asamba_demon, samba127 }  
  WinsSupport = 1  
  DomainMaster = 1  
)
```


Service and application agents

This chapter contains the following agents:

- [“Apache Web server agent”](#) on page 140
- [“Application agent”](#) on page 151
- [“Process agent”](#) on page 160
- [“ProcessOnOnly agent”](#) on page 164
- [“Zone agent”](#) on page 167
- [“LDom agent”](#) on page 170

About the service and application agents

Use service and application agents to provide high availability for application and process-related resources.

Apache Web server agent

The Apache Web server agent brings an Apache Server online, takes it offline, and monitors its processes. The Apache Web server agent consists of resource type declarations and agent scripts. You use the Apache Web server agent, in conjunction with other agents, to make an Apache Web server highly available. This agent supports the Apache HTTP server 1.3, 2.0, and 2.2. It also supports the IBM HTTP Server 1.3 and 2.0.

This agent can detect when an Apache Web server is brought down gracefully by an administrator. When Apache is brought down gracefully, the agent does not trigger a resource fault even though Apache is down.

Note: The Apache agent requires an IP resource for operation.

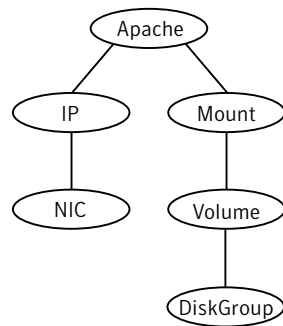
For more information regarding this agent:

See [“Apache Web server notes”](#) on page 147.

Dependencies

This type of resource depends on IP and Mount resources.

Figure 5-1 Sample service group for the Apache Web server agent



Agent functions

Online	Starts an Apache server by executing the httpdDir/httpd program with the appropriate arguments. When you specify a file with the EnvFile attribute, the file is sourced before the agent executes the httpd command.
Offline	To stop the Apache HTTP server, the agent: ■ Executes the httpdDir/httpd program with the appropriate arguments (Apache v2.0), or ■ Sends a TERM signal to the HTTP Server parent process (Apache v1.3). When you specify a file with the EnvFile attribute, the file is sourced before the agent executes the httpd command.
Monitor	Monitors the state of the Apache server. First it checks for the processes, next it can perform an optional state check.
Clean	Removes the Apache HTTP server system resources that might remain after a server fault or after an unsuccessful attempt to online or offline. These resources include the parent httpd daemon and its child daemons.
Action	checkconffile.vfd Checks for the existence of the Apache configuration file and the existence of the directory that contains the httpd binary that is used during start up. For a local installation, if the config file or HttpdDir is not found, make sure that it exists on the failover node.

State definitions

ONLINE	Indicates that the Apache server is running.
OFFLINE	Indicates that the Apache server is not running. Can also indicate that the administrator has stopped the Web server gracefully. Note that the agent uses the PidFile attribute for intentional offline detection.
UNKNOWN	Indicates that a problem exists with the configuration.

Attributes

Table 5-1 Required attributes

Required attribute	Description
ConfigFile	Full path and file name of the main configuration file for the Apache server. Type and dimension: string-scalar Example: "/apache/server1/conf/httpd.conf"
httpdDir	Full path of the directory to the httpd binary file Type and dimension: string-scalar Example: "/apache/server1/bin"
SecondLevelMonitor	Enables second-level monitoring for the resource. Second-level monitoring is a deeper, more thorough state check of the Apache HTTP server. An HTTP GET request on the Web server's root directory performs the monitoring. Valid attribute values are 1 (true) and 0 (false). Specifying this attribute is required. Type and dimension: boolean-scalar Default: 0 Example: "1"
ResLogLevel	Controls the agent's logging detail for a specific instance of a resource. Values are: ■ ERROR: Logs error messages. ■ WARN: Logs error and warning messages. ■ INFO: Logs error, warning, and informational messages. ■ TRACE: Logs error, warning, informational, and trace messages. Trace logging is verbose. Use for initial configuration or troubleshooting. Type and dimension: string-scalar Default: INFO Example: "TRACE"

Table 5-1 Required attributes

Required attribute	Description
PidFile	This attribute is required when you want to enable the detection of a graceful shutdown outside of VCS control. See “PidFile” on page 145.
EnvFile	This attribute may be required when you use IBM HTTP Server. See “EnvFile” on page 145.

Table 5-2 Optional attributes

Optional attribute	Description
DirectiveAfter	A list of directives that httpd processes after reading the configuration file. Type and dimension: string-association Example: DirectiveAfter{} = { KeepAlive=On }
DirectiveBefore	A list of directives that httpd processes before it reads the configuration file. Type and dimension: string-association Example: DirectiveBefore{} = { User=nobody, Group=nobody }
User	Account name the agent uses to execute the httpd program. If you do not specify this value, the agent executes httpd as the root user. Type and dimension: string-scalar Example: "apache1"

Table 5-2 Optional attributes

Optional attribute	Description
EnableSSL	Set to 1 (true) to have the online agent function add support for SSL by including the option <code>-DSSL</code> in the start command. For example: <pre>/usr/sbin/httpd -f path_to_httpd.conf -k start -DSSL</pre> Where <code>path_to_httpd.conf</code> file is the path to the <code>httpd.conf</code> file. Set to 0 (false) it excludes the <code>-DSSL</code> option from the command. Type and dimension: boolean-scalar Default: 0 Example: "1"
HostName	The virtual host name that is assigned to the Apache server instance. The host name is used in second-level monitoring to establish a socket connection with the Apache HTTP server. Note: The <code>HostName</code> attribute is only required when the value of <code>SecondLevelMonitor</code> is 1 (true). Type and dimension: string-scalar Example: "web1.symantec.com"
Port	Port number where the Apache HTTP server instance listens. The port number is used in second-level monitoring to establish a socket connection with the server. Specify this attribute only if <code>SecondLevelMonitor</code> is set to 1 (true). Type and dimension: integer-scalar Default: 80 Example: "80"

Table 5-2 Optional attributes

Optional attribute	Description
EnvFile	Full path and file name of the file that is sourced before executing httpdDir/httpd. With Apache 2.0, the file <i>ServerRoot/bin/envvars</i>, which is supplied in most Apache 2.0 distributions, is commonly used to set the environment before executing httpd. Specifying this attribute is optional. If EnvFile is specified, the shell for user root must be Bourne, Korn, or C shell. This attribute may be required when you use the IBM HTTP Server if the online action fails. For example: set the EnvFile to /usr/IBM/HTTPServer/bin/envvars. Type and dimension: string-scalar Example: "/apache/server1/bin/envvars"
PidFile	The PidFile attribute sets the file to which the server records the process ID of the daemon. The value of PidFile attribute must be the absolute path where the Apache instance records the pid. This attribute is required when you want the agent to detect the graceful shutdown of the Web server. For the agent to detect the graceful shutdown of the Web server, the value of the IntentionalOffline resource type attribute must be 1 (true). Type and dimension: string-scalar Example: /var/run/httpd.pid
SharedObjDir	Full path of the directory in which the Apache HTTP shared object files are located. Specifying this attribute is optional. It is used when the HTTP Server is compiled using the SHARED_CORE rule. If you specify this attribute, the directory is passed to the -R option when executing the httpd program. Refer to the httpd man pages for more information about the -R option. Type and dimension: boolean-scalar Example: "/apache/server1/libexec"

Table 5-2 Optional attributes

Optional attribute	Description
SecondLevelTime out	The number of seconds that the monitor agent function waits on the execution of second-level monitor. If the second-level monitor program does not return to calling the monitor agent function before the SecondLevelTimeout window expires, the monitor agent function no longer blocks on the program sub-process. It does, however, report that the resource is offline. The value should be high enough to allow the second level monitor enough time to complete. The value should be less than the value of the agent's MonitorTimeout. Type and dimension: integer-scalar Default: 30

Resource type definition

```
type Apache (  
    static keylist SupportedActions = { "checkconf.vfd" }  
    static str ContainerType = Zone  
    static str ArgList[] = { ResLogLevel, State, IState, httpdDir,  
        SharedObjDir, EnvFile, PidFile, HostName, Port, User,  
        SecondLevelMonitor, SecondLevelTimeout, ConfigFile, EnableSSL,  
        DirectiveAfter, DirectiveBefore }  
    str ResLogLevel = INFO  
    str httpdDir  
    str SharedObjDir  
    str EnvFile  
    str PidFile  
    str HostName  
    int Port = 80  
    str User  
    boolean SecondLevelMonitor  
    int SecondLevelTimeout = 30  
    str ConfigFile  
    boolean EnableSSL  
    str DirectiveAfter{}  
    str DirectiveBefore{}  
    str ContainerName  
    static int IntentionalOffline = 0  
)
```

Apache Web server notes

The Apache Web server has the following notes:

- [“Tasks to perform before you use the Apache Web server agent”](#) on page 147
- [“Detecting application failure”](#) on page 148
- [“About bringing an Apache Web server online outside of VCS control”](#) on page 148
- [“About the ACC Library”](#) on page 149
- [“High Availability fire drill”](#) on page 149

Tasks to perform before you use the Apache Web server agent

Before you use this agent, perform the following tasks:

- Install the Apache server on shared or local disks.
- Ensure that you are able to start the Apache Web server outside of VCS control, with the specified parameters in the Apache configuration file (for example: /etc/apache/httpd.conf). For more information on how to start the server:
See [“About bringing an Apache Web server online outside of VCS control”](#) on page 148.
- Specify the location of the error log file in the Apache configuration file for your convenience (for example: ErrorLog /var/apache/logs/error_log).
- Verify that the floating IP has the same subnet as the cluster systems.
- If you use a port other than the default 80, assign an exclusive port for the Apache server.
- Verify that the Apache server configuration files are identical on all cluster systems.
- Verify that the Apache server does not autostart on system startup.
- Verify that `Inetd` does not invoke the Apache server.
- Install the ACC Library 4.1.04.0 (VRTSacclib) if it is not already installed. If the ACC Library needs to be installed or updated, the library and its documentation can be obtained from the agent software media.

- Remove previous versions of this agent.
- The service group has disk and network resources to support the Apache server resource.
- Assign virtual host name and port to Apache Server.

Detecting application failure

The agent provides two methods to evaluate the state of an Apache HTTP server instance. The first state check is mandatory and the second is optional.

The first check determines the state of the Apache HTTP server. The check determines the state by searching for the existence of the parent httpd daemon. It also searches for at least one child httpd daemon. If the parent process and at least one child do not exist, VCS reports the resource as offline. If they do exist, and if the agent attribute `SecondLevelMonitor` is set to true, then a socket connection is established with the Apache HTTP server using the values specified by the `Host` and `Port` agent attributes. When connected, the agent issues an HTTP request to the server to test its ability to respond. If the HTTP Server responds with a return code between 0 and 408, the agent considers the server online. If the server fails to respond or returns any other code, the agent considers the server offline.

About bringing an Apache Web server online outside of VCS control

When you bring an Apache Web server online outside of VCS control, first source its environment file. Start the server with the `-f` option so the server knows which instance to start. You can then specify additional options (such as `EnableSSL` or `SharedObjDir`) that you want the server to use at start.

To start an Apache Web server outside of VCS control

- 1 Source the environment file if required.
- 2 Start the Apache Web server. You must use the `-f` option so that the agent can distinguish different instances of the server.

```
httpdDir/httpd -f ConfigFile -k start
```

Where `httpdDir` is `/apache/v2.2/bin` `ConfigFile` is `/apache/v2.2/conf/httpd.conf`. When fully formed, the start example looks like:

```
/apache/v2.2/bin/httpd -f /apache/v2.2/conf/httpd.conf -k start
```

- 3 Specify additional options such as `EnableSSL` or `SharedObjDir` that you want to use when you start server. When you add `EnableSSL` to the command, it resembles:

```
httpdDir/httpd -f ConfigFile -k start -DSSL
```

About the ACC Library

The agent functions for the Apache HTTP server depend on a set of Perl modules that are known as the ACC Library. The ACC Library contains the common, reusable functions that perform tasks such as process identification, logging, and system calls.

When you install the ACC library in a VCS environment, you must install the ACC library package before you install the agent.

To install or update the ACC library package, locate the library and related documentation on the agent disc and in the compressed agent tar file.

High Availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node. These discrepancies might prevent a service group from going online on a specific node. For Apache resources, when the Apache Web server is installed locally, the high availability fire drill checks for the validity of these attributes:

- ConfigFile
- httpdDir

For more information about using the high availability fire drill see the *Veritas Cluster Server User's Guide*.

Sample configurations

```
group ApacheG1 (
    SystemList = { host1 = 0, host2 = 1 }
)

Apache httpd_server (
    Critical = 0
    httpdDir = "/apache/bin"
    HostName = vcssl1
    Port = 8888
    User = root
    SecondLevelMonitor = 1
    ConfigFile = "/apache/conf/httpd.conf"
)

DiskGroup Apache_dg (
    Critical = 0
    DiskGroup = apc1
)
```

```
IP Apache_ip (  
    Critical = 0  
    Device = bge0  
    Address = "11.123.99.168"  
    NetMask = "255.255.254.0"  
)  
  
Mount Apache_mnt (  
    Critical = 0  
    MountPoint = "/apache"  
    BlockDevice = "/dev/vx/dsk/apc1/apcvol1"  
    FSType = vxfs  
    FsckOpt = "-y"  
)  
  
Apache_mnt requires Apache_dg  
httpd_server requires Apache_mnt  
httpd_server requires Apache_ip
```

Application agent

The Application agent brings applications online, takes them offline, and monitors their status. Use it to specify different executables for the online, offline, and monitor routines for different programs. The executables must exist locally on each node. You can use this agent to provide high availability for applications that do not have custom agents.

An application runs in the default context of root. Specify the user name to run an application in a user context.

You can monitor the application in the following ways:

- Use the monitor program
- Specify a list of processes
- Specify a list of process ID files
- Any combination of the above

High availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node. These discrepancies might prevent a service group from going online on a specific node. For Application resources, the high availability fire drill checks for:

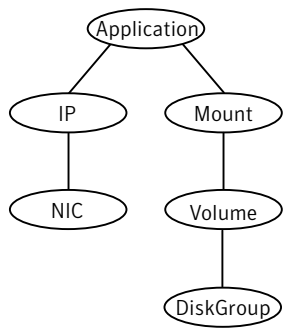
- The availability of the specified program
- Execution permissions for the specified program
- The existence of the specified user on the host
- The existence of the same binary on all nodes

For more information about using the high availability fire drill see the *Veritas Cluster Server User's Guide*.

Dependencies

Depending on how you plan to use it, this type of resource can depend on IP, IPMultiNIC, and Mount resources.

Figure 5-2 Sample service group for an Application resource



Agent functions

Online	Runs the StartProgram attribute with the specified parameters in the context of the specified user.
Offline	Runs the StopProgram attribute with the specified parameters in the context of the specified user.
Monitor	If you specify the MonitorProgram attribute, the agent executes the user-defined MonitorProgram in the user-specified context. If you specify the PidFiles attribute, the routine verifies that the process ID that is found in each listed file is running. If you specify the MonitorProcesses attribute, the routine verifies that each listed process is running in the context you specify. Use any combination among these attributes (MonitorProgram, PidFiles, or MonitorProcesses) to monitor the application. If any one the processes that are specified in either PidFiles or MonitorProcesses is determined not to be running, the monitor returns OFFLINE. If the process terminates ungracefully, the monitor returns OFFLINE and failover occurs.
Clean	Terminates processes specified in PidFiles or MonitorProcesses. Ensures that only those processes (that are specified in the MonitorProcesses attribute) running with the user ID specified in the User attribute are killed. If the CleanProgram is defined, the agent executes the CleanProgram.

State definitions

ONLINE	Indicates that all processes that are specified in the PidFiles and the MonitorProcesses attribute are running and that the MonitorProgram returns ONLINE.
OFFLINE	Indicates that at least one process that are specified in the PidFiles attribute or MonitorProcesses is not running, or that the MonitorProgram returns OFFLINE.
UNKNOWN	Indicates an indeterminable application state or invalid configuration.

Attributes

Table 5-3 Required attributes

Required attribute	Description
StartProgram	The executable, created locally on each node, which starts the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. For applications running in Solaris 10 zones, use the path as seen from the non-global zone. Note: Do not use the opening and closing ({}) brace symbols in this string. Type and dimension: string-scalar Example: "/usr/sbin/samba start"
StopProgram	The executable, created locally on each node, which stops the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. For applications running in Solaris 10 zones, use the path as seen from the non-global zone. Note: Do not use the opening and closing ({}) brace symbols in this string. Type and dimension: string-scalar Example: "/usr/sbin/samba stop"
At least one of the following attributes: ■ MonitorProcesses ■ MonitorProgram ■ PidFiles	See " Optional attributes " on page 155.

Table 5-4 Optional attributes

Optional attribute	Description
CleanProgram	The executable, created locally on each node, which forcibly stops the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. For applications running in Solaris 10 zones, use the path as seen from the non-global zone. Type and dimension: string-scalar Example: <code>"/usr/sbin/samba force stop"</code>
ContainerName	Non-global zone support for Solaris 10 and later. Defines the name of the non-global zone. Type and dimension: string-scalar Example: <code>"zone1"</code>
ContainerType	Do not change. For internal use only.
MonitorProcesses	A list of processes that you want monitored and cleaned. Each process name is the name of an executable. Provide the full path name of the executable if the agent uses that path to start the executable. The process name must be the full command line argument that the <code>ps -u user -o args</code> command displays for the process. Type and dimension: string-vector Example: <code>"nmbd"</code>

Table 5-4 Optional attributes

Optional attribute	Description
MonitorProgram	The executable, created locally on each node, which monitors the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. For applications running in Solaris 10 zones, use the path as seen from the non-global zone. MonitorProgram can return the following VCSAgResState values: OFFLINE value is 100; ONLINE values range from 101 to 110 (depending on the confidence level); 110 equals confidence level of 100%. Any other value = UNKNOWN. Note: Do not use the opening and closing ({}) brace symbols in this string. Type and dimension: string-scalar Example: "/usr/local/bin/sambaMonitor all"

Table 5-4 Optional attributes

Optional attribute	Description
PidFiles	A list of PID (process ID) files that contain the PID of the processes that you want monitored and cleaned. These are application generated files. Each PID file contains one monitored PID. Specify the complete path of each PID file in the list. For applications running in Solaris 10 non-global zones, include the zone root path in the PID file's path—the global zone's absolute path. The process ID can change when the process restarts. If the application takes time to update the PID file, the agent's monitor function may return an incorrect result. If incorrect results occur, increase the ToleranceLimit in the resource definition. Type and dimension: string-vector Example: <code>"/var/lock/samba/smbd.pid"</code> Example in a global zone for Solaris 10: <code>"/var/lock/samba/smbd.pid"</code> Example in a non-global zone for Solaris 10: <code>"\$zoneroot/var/lock/samba/smbd.pid"</code> Where the <i>\$zoneroot</i> is the root directory of the non-global zone, as seen from the global zone.
User	The user ID for running StartProgram, StopProgram, MonitorProgram, and CleanProgram. The processes that are specified in the MonitorProcesses list must run in the context of the specified user. Monitor checks the processes to make sure they run in this context. Type and dimension: string-scalar Default: root

Resource type definition

```
type Application (  
    static keylist SupportedActions = { "program.vfd", "user.vfd",  
    "cksum.vfd", getcksum }  
    static str ContainerType = Zone  
    static str ArgList[] = { User, StartProgram, StopProgram,  
    CleanProgram, MonitorProgram, PidFiles, MonitorProcesses }  
    str User  
    str StartProgram  
    str StopProgram  
    str CleanProgram  
    str MonitorProgram  
    str PidFiles[]  
    str MonitorProcesses[]  
    str ContainerName  
)
```

Sample configurations

Configuration 1

In this example, you configure the executable samba as StartProgram and StopProgram, with start and stop specified as command line arguments respectively. Configure the agent to monitor two processes: a process that the smbd.pid specifies and the process nmbd.

```
Application samba_app (  
    User = "root"  
    StartProgram = "/usr/sbin/samba start"  
    StopProgram = "/usr/sbin/samba stop"  
    PidFiles = { "/var/lock/samba/smbd.pid" }  
    MonitorProcesses = { "nmbd" }  
)
```

Configuration 2

In this example, since no user is specified, it uses the root user. The executable samba starts and stops the application using start and stop as the command line arguments. The executable sambaMonitor monitors the application and uses all as its command line argument. The agent also monitors the smbd and nmbd processes.

```
Application samba_app2 (  
    StartProgram = "/usr/sbin/samba start"  
    StopProgram = "/usr/sbin/samba stop"  
    CleanProgram = "/usr/sbin/samba force stop"  
    MonitorProgram = "/usr/local/bin/sambaMonitor all"  
    MonitorProcesses = { "smbd", "nmbd" }  
)
```

Configuration 3 for Solaris 10

In this example, configure a resource in a non-global zone: zone1. The ZonePath of zone1 is /zone1/root. Configure the executable samba as StartProgram and StopProgram, with start and stop specified as command line arguments respectively. Configure the agent to monitor two processes: a process that the smbd.pid specifies and the process nmbd.

```
Application samba_app (  
    StartProgram = "/usr/sbin/samba start"  
    StopProgram = "/usr/sbin/samba stop"  
    PidFiles = { "/zone1/root/var/lock/samba/smbd.pid" }  
    MonitorProcesses = { "nmbd" }  
)
```

Process agent

The Process agent starts, stops, and monitors a process that you specify. You can use the agent to make a process highly available or to monitor it.

High availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node; discrepancies that might prevent a service group from going online on a specific node. For Process resources, the high availability fire drill checks for:

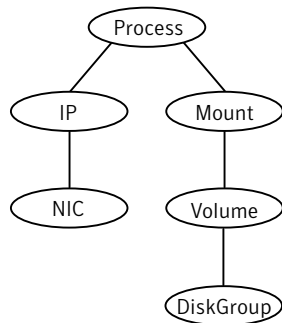
- The existence of the specified process
- Execution permissions for the specified process
- The existence of a binary executable for the specified process
- The existence of the same binary on all nodes

For more information about using the high availability fire drill see the *Veritas Cluster Server User's Guide*.

Dependencies

Depending on the context, this type of resource can depend on IP, IPMultiNIC, and Mount resources.

Figure 5-3 Sample service group for a Process resource



Agent functions

Online	Starts the process with optional arguments.
Offline	Terminates the process with a SIGTERM. If the process does not exit, a SIGKILL is sent.
Monitor	Checks to see if the process is running by scanning the process table for the name of the executable pathname and argument list.
Clean	Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

State definitions

ONLINE	Indicates that the specified process is running in the specified user context. For Solaris 10, the process can run in global and non-global zones when you specify the ContainerName attribute. The agent only reports the process as online if the value configured for PathName attribute exactly matches the process listing from the ps output.
OFFLINE	Indicates that the specified process is not running in the specified user context.
FAULTED	Indicates that the process has terminated unexpectedly.
UNKNOWN	Indicates that the agent can not determine the state of the process.

Attributes

Table 5-5 Required attribute

Required attribute	Description
PathName	Complete pathname to access an executable program. This path includes the program name. If a script controls the process, the PathName defines the complete path to the shell. This attribute must not exceed 80 characters. Type and dimension: string-scalar Example: "/usr/lib/sendmail"

Table 5-6 Optional attributes

Optional attribute	Description
Arguments	Passes arguments to the process. If a script controls the process, the script is passed as an argument. Separate multiple arguments with a single space. A string cannot accommodate more than one space between arguments, nor allow for leading or trailing whitespace characters. This attribute must not exceed 80 characters. Type and dimension: string-scalar Example: "bd -q1h"
ContainerName	Non-global zone support for Solaris 10 and above. Defines the name of the non-global zone. Type and dimension: string-scalar Example: "zone1"
ContainerType	Do not change. For internal use only.

Resource type definition

```

type Process (
    static keylist SupportedActions = { "program.vfd", getcksum }
    static str ContainerType = Zone
    static str ArgList[] = { ContainerName, PathName, Arguments }
    str ContainerName
    str PathName
    str Arguments
)

```

Sample configurations

Configuration 1

```

Process usr_lib_sendmail (
    PathName = "/usr/lib/sendmail"
    Arguments = "bd qlh"
)

```

Configuration 2

```

include "types.cf"
cluster ProcessCluster (
    .
    .
    .
    group ProcessGroup (
        SystemList = { sysa, sysb }
        AutoStartList = { sysa }
    )

    Process Process1 (
        PathName = "/usr/local/bin/myprog"
        Arguments = "arg1 arg2"
    )

    Process Process2 (
        PathName = "/bin/csh"
        Arguments = "/tmp/funscript/myscript"
    )

    // resource dependency tree
    //
    //     group ProcessGroup
    //     {
    //         Process Process1
    //         Process Process2
    //     }

```

ProcessOnOnly agent

The ProcessOnOnly agent starts and monitors a process that you specify. You can use the agent to make a process highly available or to monitor it. This resource's Operation value is OnOnly.

VCS uses this agent internally to mount security processes in a secure cluster.

Dependencies

No child dependencies exist for this resource.

Agent functions

Online	Starts the process with optional arguments.
Monitor	Checks to see if the process is alive by scanning the process table for the name of the executable pathname and argument list.
Clean	Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

State definitions

ONLINE	Indicates that the specified process is running. For Solaris 10, the process can run in global and non-global zones when you specify the ContainerName attribute. The agent only reports the process as ONLINE if the value configured for PathName attribute exactly matches the process listing from the ps output.
FAULTED	Indicates that the process has unexpectedly terminated.
UNKNOWN	Indicates that the agent can not determine the state of the process.

Attributes

Table 5-7 Required attributes

Required attribute	Description
PathName	Defines complete pathname to access an executable program. This path includes the program name. If a process is controlled by a script, the PathName defines the complete path to the shell. Pathname must not exceed 80 characters. The value configured for this attribute needs to match the process listing from the ps output for the agent to display as ONLINE. Type and dimension: string-scalar Example: <code>"/usr/lib/nfs/nfsd"</code>

Table 5-8 Optional attributes

Optional attribute	Description
Arguments	Passes arguments to the process. If a process is controlled by a script, the script is passed as an argument. Multiple arguments must be separated by a single space. A string cannot accommodate more than one space between arguments, nor allow for leading or trailing whitespace characters. Arguments must not exceed 80 characters (total). Type and dimension: string-scalar Example: <code>"- a 8"</code>
ContainerName	Non-global zone support for Solaris 10 and above. Defines the name of the non-global zone. Type and dimension: string-scalar Example: <code>"zone1"</code>
ContainerType	Do not change. For internal use only.

Table 5-8 Optional attributes

Optional attribute	Description
IgnoreArgs	A flag that indicates whether monitor ignores the argument list. ■ If the value is 0, it checks the process pathname and argument list. ■ If the value is 1, it only checks for the executable pathname and ignores the rest of the argument list. Type and dimension: boolean-scalar Default: 0

Resource type definition

```
type ProcessOnOnly (  
    static str ContainerType = Zone  
    static str ArgList[] = { ContainerName, IgnoreArgs, PathName,  
        Arguments }  
    static str Operations = OnOnly  
    str ContainerName  
    boolean IgnoreArgs = 0  
    str PathName  
    str Arguments  
)
```

Sample configurations

```
group VxSS (  
    SystemList = { north = 0, south = 1 }  
    Parallel = 1  
    OnlineRetryLimit = 3  
    OnlineRetryInterval = 120  
)  
  
Phantom phantom_vxss (  
)  
  
ProcessOnOnly vxatd (  
    IgnoreArgs = 1  
    PathName = "/opt/VRTSat/bin/vxatd"  
)  
  
// resource dependency tree  
//  
// group VxSS
```

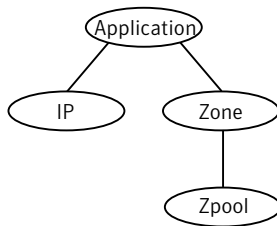
Zone agent

The Zone agent brings online, takes offline, monitors, and cleans Solaris 10 zones. You can use the agent to make zones highly available and to monitor them.

Dependencies

No dependencies exist for the Zone resource.

Figure 5-4 Sample service group for a Zone resource



Agent functions

Online	Brings a Solaris 10 zone up and running.
Offline	Takes a Solaris 10 zone down gracefully.
Monitor	Checks if the specified zone is up and running.
Clean	A more forceful method for halting a Solaris 10 zone.

Attributes

Table 5-9 Required attributes

Required attribute	Description
ZoneName	Name of the zone Type and dimension: string-scalar Example: "localzone1"

Table 5-10 Optional attributes

Optional attribute	Description
ShutDownGracePeriod	Specifies the interval in seconds from the Offline action to the execution of the shutdown within the zone. Type and dimension: integer-scalar Default: 0 Example: "10"

Resource type definition

```
type Zone (  
  static str ArgList[] = { ZoneName, ShutdownGracePeriod }  
  str ZoneName  
  int ShutdownGracePeriod  
)
```

Sample configuration

```
Zone myzone (  
    ZoneName = "localzone1"  
)
```

Configuration for Solaris 10

In this example, configure a resource in a non-global zone: localzone1. The ZonePath of localzone1 is /zone1/root. Configure the executable samba as StartProgram and StopProgram, with start and stop specified as command line arguments respectively. Configure the agent to monitor two processes: a process specified by the pid smbd.pid, and the process nmbd.

```
Application samba_app (  
    StartProgram = "/usr/sbin/samba start"  
    StopProgram = "/usr/sbin/samba stop"  
    PidFiles = { "/localzone1/root/var/lock/samba/smbd.pid" }  
    MonitorProcesses = { "nmbd" }  
    ContainerName = "localzone1"  
)
```

LDom agent

The LDom agent brings logical domains (LDoms) online, takes them offline, and monitors them. You can use this agent to monitor LDoms, and to make them highly available.

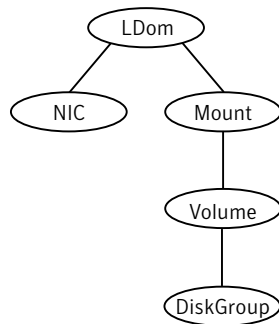
Limitations

Due to a Sun limitation, the LDom agent requires at least two VCPUs per LDom.

Dependencies

The LDom resource depends on the NIC resource. It can depend on the Volume or Mount resources in different environments.

Figure 5-5 Sample service group for an LDom resource that monitors an image file



Network resources

Use the NIC agent to monitor the network adapter for the LDom.

Storage resources

- Veritas Volume Manager (VxVM) exposed volumes
Use the Volume and DiskGroup agents to monitor a VxVM volume.
- Image file
Use the Mount, Volume, and DiskGroup agents to monitor an image file.
- Primary network interface
Use the NIC agent to monitor the primary network interface, whether it is virtual or physical.

Agent functions

Online	Starts the LDom.
Offline	Stops the LDom.
Monitor	Monitors the status of the LDom.
Clean	Stops the LDom forcefully.

State definitions

ONLINE	Indicates that the LDom is up and running.
OFFLINE	Indicates that the LDom is down.
FAULTED	Indicates that the LDom is down when VCS expects it to up and running. 100% CPU utilization of the LDom is a fault.
UNKNOWN	Indicates the agent cannot determine the LDom's state. A configuration problem likely exists in the VCS resource or the LDom.

Attributes

Table 5-11 Required attributes

Required attribute	Description
LDomName	The name of the LDom that you want to monitor. Type-dimension: string-scalar Default: n/a Example: "ldom1"

Table 5-12 Optional attributes

Required attribute	Description
CfgFile	The absolute location of the XML file that contains the LDom configuration. The file must be present locally on all those nodes where you want VCS to create the LDom. The online agent function uses this file to create LDoms as necessary. Refer to the <code>ldm(1M)</code> man page for information on this file. Type-dimension: string-scalar Default: n/a Example: "/root/ldom-cfg/ldom1.xml"
NumCPU	The number of virtual CPUs that you want to attach to the LDom when it is online. If you set this attribute to a positive value, the agent detaches all of the VCPUs when the service group goes offline. Do not reset this value to 0 after setting it to 1. Type-dimension: integer-scalar Default: 0

Resource type definition

```
type LDom (  
    static keylist RegList = { NumCPU }  
    static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"  
    static str ArgList[] = { LDomName, CfgFile, NumCPU}  
    int NumCPU  
    str LDomName  
    str CfgFile  
)
```

Sample configuration

```
LDom ldom1 (  
    LDomName = "ldom1"  
)
```


Infrastructure and support agents

This chapter contains the following agents:

- “[NotifierMngr agent](#)” on page 176
- “[VRTSWebApp agent](#)” on page 183
- “[Proxy agent](#)” on page 186
- “[Phantom agent](#)” on page 190
- “[RemoteGroup agent](#)” on page 192

About the infrastructure and support agents

Use the infrastructure and support agents to monitor Veritas components and VCS objects.

NotifierMngr agent

Starts, stops, and monitors a notifier process, making it highly available. The notifier process manages the reception of messages from VCS and the delivery of those messages to SNMP consoles and SMTP servers. See the *Veritas Cluster Server User's Guide* for a description of types of events that generate notification. See the `notifier(1)` manual page to configure notification from the command line.

You cannot dynamically change the attributes of the NotifierMngr agent using the `hares -modify` command. Changes made using this command are only effective after restarting the notifier.

Dependency

The NotifierMngr resource can depend on the NIC resource.

Agent functions

Online	Starts the notifier process with its required arguments.
Offline	VCS sends a <code>SIGABORT</code> . If the process does not exit within one second, VCS sends a <code>SIGKILL</code> .
Monitor	Monitors the notifier process.
Clean	Sends <code>SIGKILL</code> .

State definitions

ONLINE	Indicates that the Notifier process is running.
OFFLINE	Indicates that the Notifier process is not running.
UNKNOWN	Indicates that the user did not specify the required attribute for the resource.

Attributes

Table 6-1 Required attributes

Required attribute	Description
SnmpConsoles	Specifies the machine names of the SNMP managers and the severity level of the messages to be delivered. The severity levels of messages are Information, Warning, Error, and SevereError. Specifying a given severity level for messages generates delivery of all messages of equal or higher severity. Note: SnmpConsoles is a required attribute if SmtServer is not specified; otherwise, SnmpConsoles is an optional attribute. Specify both SnmpConsoles and SmtServer if desired. Type and dimension: string-association Example: "172.29.10.89" = Error, "172.29.10.56" = Information
SmtServer	Specifies the machine name of the SMTP server. Note: SmtServer is a required attribute if SnmpConsoles is not specified; otherwise, SmtServer is an optional attribute. You can specify both SmtServer and SnmpConsoles if desired. Type and dimension: string-scalar Example: "smtp.your_company.com"

Table 6-2 Optional attributes

Optional attribute	Description
EngineListeningPort	Change this attribute if the VCS engine is listening on a port other than its default port. Type and dimension: integer-scalar Default: 14141

Table 6-2 Optional attributes

Optional attribute	Description
MessagesQueue	Size of the VCS engine's message queue. Minimum value is 30. Type and dimension: integer-scalar Default: 30
NotifierListeningPort	Any valid, unused TCP/IP port number. Type and dimension: integer-scalar Default: 14144
SmtplibFromPath	Set to a valid email address, if you want the notifier to use a custom email address in the FROM: field. Type and dimension: string-scalar Example: "usera@example.com"
SmtplibRecipients	Specifies the email address where SMTP sends information and the severity level of the messages. The severity levels of messages are Information, Warning, Error, and SevereError. Specifying a given severity level for messages indicates that all messages of equal or higher severity are received. Note: SmtplibRecipients is a required attribute if you specify SmtplibServer. Type and dimension: string-association Example: "james@symantec.com" = SevereError, "admin@symantec.com" = Warning

Table 6-2 Optional attributes

Optional attribute	Description
SmtpReturnPath	Set to a valid email address, if you want the notifier to use a custom email address in the Return-Path: <> field. If the mail server specified in SmtpServer does not support VRFY, then you need to set the SmtpVrfyOff to 1 in order for the SmtpReturnPath value to take effect. Type and dimension: string-scalar Example: "usera@example.com"
SmtpServerTimeout	This attribute represents the time in seconds notifier waits for a response from the mail server for the SMTP commands it has sent to the mail server. This value can be increased if you notice that the mail server is taking a longer duration to reply back to the SMTP commands sent by notifier. Type and dimension: integer-scalar Default: 10
SmtpServerVrfyOff	Set this value to 1 if your mail server does not support SMTP VRFY command. If you set this value to 1, the notifier does not send a SMTP VRFY request to the mail server specified in SmtpServer attribute while sending emails. Type and dimension: boolean-scalar Default: 0
SnmpCommunity	Specifies the community ID for the SNMP manager. Type and dimension: string-scalar Default: public

Table 6-2 Optional attributes

Optional attribute	Description
SnmpdTrapPort	Port on the SNMP console machine where SNMP traps are sent. If you specify more than one SNMP console, all consoles use this value. Type and dimension: integer-scalar Default: 162

Resource type definition

```
type NotifierMngr (  
  static int RestartLimit = 3  
  static str ArgList[] = { EngineListeningPort, MessagesQueue,  
    NotifierListeningPort, SnmpdTrapPort, SnmpCommunity,  
    SnmpConsoles, Smtperver, SmtperverVrfyOff, SmtperverTimeout,  
    SmtperreturnPath, SmtperfromPath, Smtperrecipients }  
  int EngineListeningPort = 14141  
  int MessagesQueue = 30  
  int NotifierListeningPort = 14144  
  int SnmpdTrapPort = 162  
  str SnmpCommunity = "public"  
  str SnmpConsoles{}  
  str Smtperver  
  boolean SmtperverVrfyOff = 0  
  int SmtperverTimeout = 10  
  str SmtperreturnPath  
  str SmtperfromPath  
  str Smtperrecipients{}  
)
```

Sample configuration

In the following configuration, the NotifierMngr agent is configured to run with two resource groups: NicGrp and Grp1. NicGrp contains the NIC resource and a Phantom resource that enables VCS to determine the online and offline status of the group. See the Phantom agent for more information on verifying the status of groups that only contain OnOnly or Persistent resources such as the NIC resource. You must enable NicGrp to run as a parallel group on both systems.

Grp1 contains the NotifierMngr resource (ntfr) and a Proxy resource (nicproxy), configured for the NIC resource in the first group.

In this example, NotifierMngr has a dependency on the Proxy resource.

Note: Only one instance of the notifier process can run in a cluster. The process cannot run in a parallel group.

The NotifierMngr resource sets up notification for all events to the SNMP console `snmpserv`. In this example, only messages of SevereError level are sent to the SMTP server (`smtp.example.com`), and the recipient (`vcadmin@example.com`).

Configuration

```
system north

system south

group NicGrp (
    SystemList = { north, south}
    AutoStartList = { north }
    Parallel = 1
)

Phantom my_phantom (

NIC    NicGrp_en0 (
    Enabled = 1
    Device  = en0
    NetworkType = ether
)

group Grp1 (
    SystemList = { north, south }
    AutoStartList = { north }
)
```

```
Proxy nicproxy(  
  TargetResName = "NicGrp_en0"  
)  
  
NotifierMngr ntfr (  
  SnmpConsoles = { snmpserv = Information }  
  SmtServer = "smtp.example.com"  
  SmtRecipients = { "vcsadmin@example.com" = SevereError }  
)  
  
ntfr requires nicproxy  
  
// resource dependency tree  
//  
//      group Grp1  
//      {  
//      NotifierMngr ntfr  
//      {  
//      Proxy nicproxy  
//      }  
//      }  
//      }
```

VRTSWebApp agent

Brings Web applications online, takes them offline, and monitors their status. This agent is used to monitor the Web consoles of various Symantec products, such as the Cluster Management Console.

The application is a Java Web application conforming to the Servlet Specification 2.3/JSP Specification 1.2 and runs inside of the Java Web server installed as a part of the VRTSweb package.

Agent functions

Online	Starts the Web application with the specified parameters. If the Web server is not already running, it first starts the server.
Offline	Removes the Web application from the Web server. If no other Web application is running, it shuts down the Web server.
Monitor	Checks if the specified Web application is currently running inside the Web server. If the application is running, monitor reports ONLINE. If the application is not running, monitor reports OFFLINE.
Clean	Removes the Web application from the Web server. If no other Web application is running, it shuts down the Web server.

State definitions

ONLINE	Indicates that the Web application is running.
OFFLINE	Indicates that the Web application is not running.
UNKNOWN	Indicates that the agent could not determine the state of the resource or that the resource attributes are invalid.

Attributes

Table 6-3 Required attributes

Required attribute	Description
AppName	Name of the application as it appears in the Web server. Type and dimension: string-scalar Example: "cmc"
InstallDir	Path to the Web application installation. You must install the Web application as a .war file with the same name as the AppName parameter. Point this attribute to the directory that contains this .war file. Type and dimension: string-scalar Example: If the AppName is cmc and InstallDir is: /opt/VRTSweb/VERITAS, the agent constructs the path for the Web application as : /opt/VRTSweb/VERITAS/cmc.war
TimeForOnline	The time the Web application takes to start after loading it into the Web server. This parameter is returned as the exit value of the online script, which inform VCS of the time it needs to wait before calling monitor on the Web application resource. This attribute is typically at least five seconds. Type and dimension: integer-scalar Example: "5"

Resource type definition

```
type VRTSWebApp (  
    static str ArgList[] = { AppName, InstallDir, TimeForOnline }  
    str AppName  
    str InstallDir  
    int TimeForOnline  
    static int NumThreads = 1  
)
```

Sample configuration

```
VRTSWebApp VCSweb (  
  AppName = "cmc"  
  InstallDir = "/opt/VRTSweb/VERITAS"  
  TimeForOnline = 5  
)
```

Proxy agent

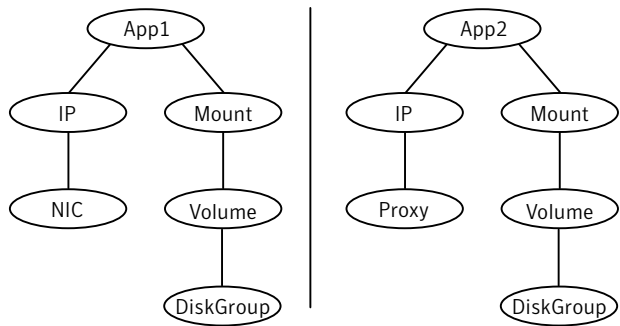
The Proxy agent mirrors the state of another resource on a local or remote system. It provides a means to specify and modify one resource and have its state reflected by its proxies. You can use the agent when you need to replicate the status of a resource.

A Proxy resource can only point to None or OnOnly type of resources, and can reside in a failover/parallel group. A target resource and its proxy cannot be in the same group.

Dependencies

No dependencies exist for the Proxy resource.

Figure 6-1 Sample service group for an Proxy resource



Agent functions

Monitor	Determines status based on the target resource status.
---------	--

Attributes

Table 6-4 Required attribute

Required attribute	Description
TargetResName	Name of the target resource that the Proxy resource mirrors. The target resource must be in a different resource group than the Proxy resource. Type and dimension: string-scalar Example: "tmp_VRTSvc_file1"

Table 6-5 Optional attribute

Optional attribute	Description
TargetSysName	Mirrors the status of the TargetResName attribute on systems that the TargetSysName variable specifies. If this attribute is not specified, the Proxy resource assumes the system is local. Type and dimension: string-scalar Example: "sysa"

Resource type definition

```
type Proxy (
    static str ArgList[] = { TargetResName, TargetSysName,
        "TargetResName:Probed", "TargetResName:State" }
    static int OfflineMonitorInterval = 60
    static str Operations = None
    str TargetResName
    str TargetSysName
)
```

Sample configurations

Configuration 1

The proxy resource mirrors the state of the resource tmp_VRTSvcs_file1 on the local system.

```
Proxy proxy1 (
    TargetResName = "tmp_VRTSvcs_file1"
)
```

Configuration 2

The proxy resource mirrors the state of the resource tmp_VRTSvcs_file1 on sysa.

```
Proxy proxy1(
    TargetResName = "tmp_VRTSvcs_file1"
    TargetSysName = "sysa"
)
```

Configuration 3

The proxy resource mirrors the state of the resource mnica on the local system; note that target resource is in grp1, and the proxy in grp2; a target resource and its proxy cannot be in the same group.

```
group grp1 (
    SystemList = { sysa, sysb }
    AutoStartList = { sysa }
)

MultiNICA mnica (
    Device@sysa = { le0 = "166.98.16.103", qfe3 = "166.98.16.103"
}

    Device@sysb = { le0 = "166.98.16.104", qfe3 = "166.98.16.104"
}

    NetMask = "255.255.255.0"
    ArpDelay = 5
    Options = "trailers"
    RouteOptions@sysa = "default 166.98.16.103 0"
```

```
        RouteOptions@sysb = "default 166.98.16.104 0"
    )
    IPMultiNIC ip1 (
        Address = "166.98.14.78"
        NetMask = "255.255.255.0"
        MultiNICResName = mnic
        Options = "trailers"
    )

    ip1 requires mnic

    group grp2 (
        SystemList = { sysa, sysb }
        AutoStartList = { sysa }
    )
    IPMultiNIC ip2 (
        Address = "166.98.14.79"
        NetMask = "255.255.255.0"
        MultiNICResName = mnic
        Options = "mtu m"
    )
    Proxy proxy (
        TargetResName = mnic
    )
    ip2 requires proxy
```

Phantom agent

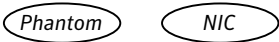
The Phantom agent enables VCS to determine the status of parallel service groups that do not include OnOff resources. Do not use the Phantom agent in failover service groups. You can use the agent to determines the state of service groups having resources of type None only.

Do not attempt manual online or offline operations on the Phantom resource or on the service group containing the Phantom resource. Doing so may result in unpredictable behavior.

Dependencies

No dependencies exist for the Phantom resource.

Figure 6-2 Sample service group for a Phantom resource



Agent functions

Monitor	Determines status based on the status of the service group.
---------	---

Attribute

Table 6-6 Attribute

Attribute	Description
Dummy	The Dummy attribute is for internal use only.

Resource type definition

```
type Phantom (  
    static str ArgList[] = { Dummy }  
    str Dummy  
)
```

Sample configurations

Configuration 1

```
Phantom boo (  
)
```

Configuration 2

The following example shows a complete main.cf, in which the FileNone resource and the Phantom resource are in the same group.

```
include "types.cf"  
  
cluster PhantomCluster  
  
system sysa  
  
system sysb  
  
group phantomgroup (  
    SystemList = { sysa, sysb }  
    AutoStartList = { sysa }  
    Parallel = 1  
)  
  
FileNone my_file_none (  
    PathName = "/tmp/file_none"  
)  
Phantom my_phantom (  
)  
  
// resource dependency tree  
//  
//     group maingroup  
//     {  
//         Phantom my_Phantom  
//         FileNone my_file_none  
//     }
```

RemoteGroup agent

The RemoteGroup agent establishes dependencies between applications that are configured on different VCS clusters. For example, you configure an Apache resource in a local cluster, and a MySQL resource in a remote cluster. In this example, the Apache resource depends on the MySQL resource. You can use the RemoteGroup agent to establish this dependency between these two resources.

With the RemoteGroup agent, you can monitor or manage a service group that exists in a remote cluster. Some points about configuring the RemoteGroup resource follow:

- For each remote service group that you want to monitor or manage, you must configure a corresponding RemoteGroup resource in the local cluster.
- Multiple RemoteGroup resources in a local cluster can manage corresponding multiple remote service groups in different remote clusters.
- You can include the RemoteGroup resource in any kind of resource or service group dependency tree.
- A combination of the state of the local service group and the state of the remote service group determines the state of the RemoteGroup resource.

Symantec supports the RemoteGroup agent when it points to a global group. The RemoteGroup agent must then map the state of the global group in the local cluster.

For more information on the functionality of this agent see the *Veritas Cluster Server User's Guide*.

Dependency

As a best practice, establish a RemoteGroup resource dependency on a NIC resource. Symantec recommends that the RemoteGroup resource not be by itself in a service group.

Agent functions

Online	Brings the remote service group online. See the “ ControlMode ” on page 195 for more information.
Offline	Takes the remote service group offline. See the “ ControlMode ” on page 195 for more information.
Monitor	Monitors the state of the remote service group. The true state of the remote service group is monitored only on the online node in the local cluster. See the “ VCSSysName ” on page 194.
Clean	If the RemoteGroup resource faults, the Clean function takes the remote service group offline. See the “ ControlMode ” on page 195 for more information.

State definitions

ONLINE	Indicates that the remote service group is either in an ONLINE or PARTIAL state.
OFFLINE	Indicates that the remote service group is in an OFFLINE or FAULTED state. The true state of the remote service group is monitored only on the online node in the local cluster.
FAULTED	Indicates that the RemoteGroup resource has unexpectedly gone offline.
UNKNOWN	Indicates that a problem exists either with the configuration or the ability of the RemoteGroup resource to determine the state of the remote service group.

Attributes

Table 6-7 Required attributes

Required attribute	Description
IpAddress	The IP address or DNS name of a node in the remote cluster. The IP address can be either physical or virtual. When configuring a virtual IP address of a remote cluster, do not configure the IP resource as a part of the remote service group. Type and dimension: string-scalar Examples: "www.example.com" or "11.183.12.214"
Port	This is a required attribute when the remote cluster listens on a port other than the default value of 14141. See “Port” on page 197.
GroupName	The name of the service group on the remote cluster that you want the RemoteGroup agent to monitor or manage. Type and dimension: string-scalar Example: "DBGrp"
VCSSysName	You must set this attribute to either the VCS system name or the ANY value. ■ ANY The RemoteGroup resource goes online if the remote service group is online on any node in the remote cluster. ■ <i>VCSSysName</i> Use the name of a VCS system in a remote cluster where you want the remote service group to be online when the RemoteGroup resource goes online. Use this to establish a one-to-one mapping between the nodes of the local and remote clusters. Type and dimension: string-scalar Example: "vcssys1" or "ANY"

Table 6-7 Required attributes

Required attribute	Description
ControlMode	Select only one of these values to determine the mode of operation of the RemoteGroup resource: MonitorOnly, OnlineOnly, or OnOff. ■ OnOff The RemoteGroup resource brings the remote service group online or takes it offline. When you set the VCSSysName attribute to ANY, the SysList attribute of the remote service group determines the node where the remote service group online. ■ MonitorOnly The RemoteGroup resource only monitors the state of the remote service group. The RemoteGroup resource cannot online or offline the remote service group. Make sure that you bring the remote service group online before you online the RemoteGroup resource. ■ OnlineOnly The RemoteGroup resource only brings the remote service group online. The RemoteGroup resource cannot take the remote service group offline. When you set the VCSSysName attribute to ANY, the SysList attribute of the remote service group determines the node where the remote service group online. Type and dimension: string-scalar

Table 6-7 Required attributes

Required attribute	Description
Username	This is the login user name for the remote cluster. When you set the ControlMode attribute to OnOff or OnlineOnly, the Username must have administrative privileges for the remote service group that you specify in the GroupName attribute. When you use the RemoteGroup Wizard to enter your username data, you need to enter your username and the domain name in separate fields. For a cluster that has the Symantec Product Authentication Service, you do not need to enter the domain name. For a secure remote cluster: ■ Local Unix user user@nodename—where the nodename is the name of the node that is specified in the IPAddress attribute. Do not set the DomainType attribute. ■ NIS or NIS+ user user@domainName—where domainName is the name of the NIS or NIS+ domain for the user. You must set the value of the DomainType attribute to either to nis or nisplus. Type and dimension: string-scalar Example: ■ For a cluster without the Symantec Product Authentication Service: "johnsmith" ■ For a secure remote cluster: "foobar@example.com"
Password	This is the password that corresponds to the user that you specify in the Username attribute. You must encrypt the password with the <code>vcseencrypt -agent</code> command. Note: Do not use the vcseencrypt utility when entering passwords from a configuration wizard or from the Cluster Management Console or the Cluster Manager (Java Console). Type and dimension: string-scalar

Table 6-8 Optional attributes

Optional attribute	Description
DomainType	For a secure remote cluster only, enter the domain type information for the specified user. For users who have the domain type unixpwd, you do not have to set this attribute. Type: string-scalar Example: "nis", "nisplus"
BrokerIp	For a secure remote cluster only. If you need the RemoteGroup agent to communicate to a specific authentication broker, set the value of this attribute to the broker's IP address. Type: string-scalar Example: "128.11.295.51"
Port	The port where the remote engine listens for requests. This is an optional attribute, unless the remote cluster listens on a port other than the default value of 14141. Type and dimension: integer-scalar Default: 14141
OfflineWaitTime	The maximum expected time in seconds that the remote service group may take to offline. VCS calls the clean function for the RemoteGroup resource if the remote service group takes a longer time to offline than the time that you have specified for this attribute. Type and dimension: integer-scalar Default: 0

Table 6-9 Type-level attributes

Type level attributes	Description
OnlineRetryLimit OnlineWaitLimit ToleranceLimit MonitorInterval AutoFailover	In case of remote service groups that take a longer time to Online, Symantec recommends that you modify the default OnlineWaitLimit and OnlineRetryLimit attributes. If you expect the RemoteGroup agent to tolerate sudden offlines of the remote service group, then modify the ToleranceLimit attribute. See the <i>Veritas Cluster Server User's Guide</i> for more information about these attributes.

Resource type definition

```
type RemoteGroup (  
  static int OnlineRetryLimit = 2  
  static int ToleranceLimit = 1  
  static str ArgList[] = { IPAddress, Port, Username, Password,  
    GroupName, VCSSysName, ControlMode, OfflineWaitTime,  
    DomainType, BrokerIp }  
  str IPAddress  
  int Port = 14141  
  str Username  
  str Password  
  str GroupName  
  str VCSSysName  
  str ControlMode  
  int OfflineWaitTime  
  str DomainType  
  str BrokerIp  
)
```

Testing agents

This chapter contains the following agents:

- [“ElifNone agent”](#) on page 200
- [“FileNone agent”](#) on page 202
- [“FileOnOff agent”](#) on page 204
- [“FileOnOnly agent”](#) on page 206

About the program support agents

Use the program support agents to provide high availability for program support resources.

ElifNone agent

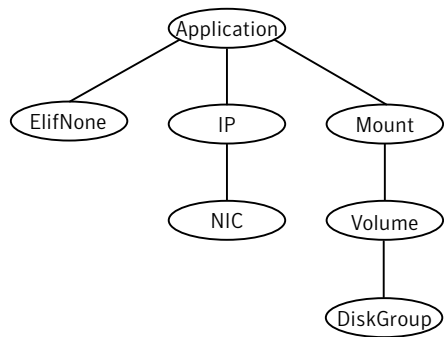
The ElifNone agent monitors a file. It checks for the file’s absence.

You can use the ElifNone agent to test service group behavior. You can also use it as an impostor resource, where it takes the place of a resource for testing.

Dependencies

No dependencies exist for the ElifNone resource.

Figure 7-1 Sample service group for an ElifNone resource



Agent function

Monitor	Checks for the specified file. If it exists, the resource faults. If it does not exist, the agent reports as ONLINE.
---------	--

Attributes

Table 7-1 Required attribute

Required attribute	Description
PathName	Specifies the complete pathname. Starts with a slash (/) preceding the file name. Type and dimension: string-scalar Example: "/tmp/file01"

Resource type definition

```
type ElifNone (  
    static str ArgList[] = { PathName }  
    static int OfflineMonitorInterval = 60  
    static str Operations = None  
    str PathName  
)
```

Sample configuration

```
ElifNone tmp_file01 (  
    PathName = "/tmp/file01"  
)
```

FileNone agent

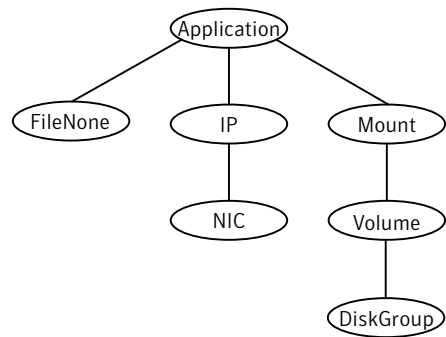
Monitors a file—checks for the file’s existence.

You can use the FileNone agent to test service group behavior. You can also use it as an “impostor” resource, where it takes the place of a resource for testing.

Dependencies

No dependencies exist for the FileNone resource.

Figure 7-2 Sample service group for an FileNone resource



Agent functions

Monitor	Checks for the specified file. If it exists, the agent reports as ONLINE. If it does not exist, the resource faults.
---------	--

Attribute

Table 7-2 Required attribute

Required attribute	Description
PathName	Specifies the complete pathname. Starts with a slash (/) preceding the file name. Type and dimension: string-scalar Example: "/tmp/file01"

Resource type definition

```
type FileNone (  
    static str ArgList[] = { PathName }  
    static int OfflineMonitorInterval = 60  
    static str Operations = None  
    str PathName  
)
```

Sample configuration

```
FileNone tmp_file01 (  
    PathName = "/tmp/file01"  
)
```

FileOnOff agent

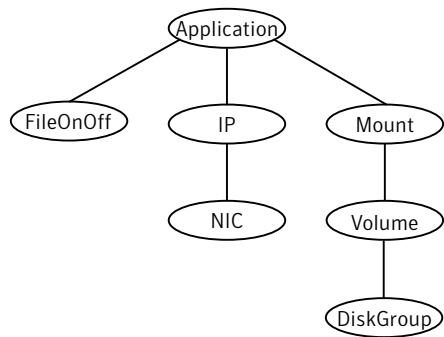
The FileOnOff agent creates, removes, and monitors files.

You can use the FileNone agent to test service group behavior. You can also use it as an “impostor” resource, where it takes the place of a resource for testing.

Dependencies

No dependencies exist for the FileOnOff resource.

Figure 7-3 Sample service group for a FileOnOff resource



Agent functions

Online	Creates an empty file with the specified name if the file does not already exist.
Offline	Removes the specified file.
Monitor	Checks for the specified file. If it exists, the agent reports as ONLINE. If it does not exist, the agent reports as OFFLINE.
Clean	Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

Attribute

Table 7-3 Required attribute

Required attribute	Description
PathName	Specifies the complete pathname. Starts with a slash (/) preceding the file name. Type and dimension: string-scalar Example: "/tmp/file01"

Resource type definition

```
type FileOnOff (  
    static str ArgList[] = { PathName }  
    str PathName  
)
```

Sample configuration

```
FileOnOff tmp_file01 (  
    PathName = "/tmp/file01"  
)
```

FileOnOnly agent

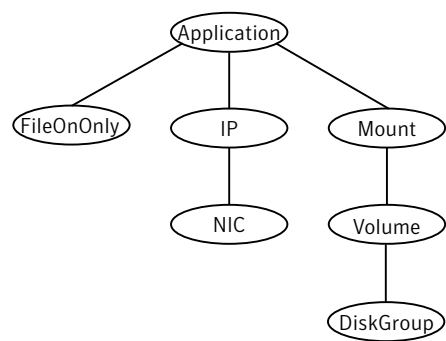
The FileOnOnly agent creates and monitors files.

You can use the FileNone agent to test service group behavior. You can also use it as an “impostor” resource, where it takes the place of a resource for testing.

Dependencies

No dependencies exist for the FileOnOnly resource.

Figure 7-4 Sample service group for a FileOnOnly resource



Agent functions

Online	Creates an empty file with the specified name, unless one already exists.
Monitor	Checks for the specified file. If it exists, the agent reports as ONLINE. If it does not exist, the resource faults.

Attribute

Table 7-4 Required attributes

Required attribute	Description
PathName	Specifies the complete pathname. Starts with a slash (/) preceding the file name. Type and dimension: string-scalar Example: "/tmp/file02"

Resource type definition

```
type FileOnOnly (  
    static str ArgList[] = { PathName }  
    static str Operations = OnOnly  
    str PathName  
)
```

Sample configuration

```
FileOnOnly tmp_file02 (  
    PathName = "/tmp/file02"  
)
```


Glossary

administrative IP address

The operating system controls these IP addresses and brings them up even before VCS brings applications online. Use them to access a specific system over the network for doing administrative tasks, for example: examining logs to troubleshoot issues, cleaning up temp files to free space, etc. Typically, you have one administrative IP address per node.

agent function

Agent functions start, stop, fault, forcibly stop, and monitor resources using scripts. Sometimes called an entry point.

base IP address

The first logical IP address, can be used as an administrative IP address.

entry point

See [agent function](#).

floating IP address

See [virtual IP address](#).

logical IP address

Any IP address assigned to a NIC.

NIC bonding

Combining two or more NICs to form a single logical NIC, which creates a fatter pipe.

operation

All agents have scripts that turn the resource on and off. Operations determine the action that the agent passes to the resource. See None operation, OnOff operation, and OnOnly operation.

None operation

For example the NIC resource. Also called persistent resource, this resource is always on. This kind of resource has no online and offline scripts, and only monitors a resource.

OnOff operation

For example the IP and Share agents--in fact most agents are OnOff. This resource has online and offline scripts. Often this type of resource does not appear in the types file because by default when a resource does not have this resource type defined, it is OnOff.

OnOnly operation

For example the NFS, FileOnOnly resources. This kind of resource has an online script, but not an offline one.

plumb

Term for enabling an IP address—used across all platforms in this guide.

test IP address

IP addresses to help determine the state of a link by sending out a ping probe to another NIC (on another system.) Requires a return ping to complete the test. Test IP addresses can be the same as base IP addresses.

virtual IP address

IP addresses that can move from one NIC to another or from one node to another. VCS fails over these IP address with your application. Sometimes called a floating IP address.

Index

Numerics

802.1Q trunking 57

A

about

- Network agents 55
- Samba agents 126

ACC library 149

agent

- modifying 20

agent functions

- Apache Web server agent 141
- Application agent 152
- DiskGroup agent 24
- DiskGroupSnap agent 32
- DNS agent 98
- ElifNone agent 200
- FileNone agent 202
- FileOnOff agent 204
- FileOnOnly agent 206
- IP agent 58
- IPMultiNIC agent 68
- IPMultiNICB agent 80
- LDom agent 170
- Mount agent 40
- MultiNICA agent 72
- MultiNICB agent 85
- NetBIOS agent 134
- NFS agent 110
- NFSRestart agent 115
- NIC agent 64
- NotifierMngr agent 176
- Phantom agent 190
- Process agent 160
- ProcessOnOnly agent 164
- Proxy agent 186
- RemoteGroup agent 192
- SambaServer agent 128
- SambaShare agent 131
- Share agent 123
- Volume agent 38

VRTSWebApp agent 183

Zone agent 167

Zpool agent 50

agents

- Apache Web server 140
- Application 151
- DiskGroup 24
- DiskGroupSnap 31
- DNS 98
- ElifNone 200
- FileNone 202
- FileOnOff 204
- FileOnOnly 206
- IP 58
- IPMultiNIC 68
- IPMultiNICB 80
- LDom 170
- Mount 40
- MultiNICA 72
- MultiNICB 85
- NetBIOS 134
- NFS 110
- NFSRestart 115
- NIC 64
- NotifierMngr 176
- Phantom 190
- Process 160
- ProcessOnOnly 164
- Proxy 186
- RemoteGroup 192
- SambaServer 128
- SambaShare 131
- Share 123
- Volume 38
- VRTSWebApp 183
- Zone 167
- Zpool 49

agents, typical functions 19

- Apache Web server agent
- ACC library 149
- agent functions 141
- attributes 142

- description 140
- detecting application failure 148
- sample configuration 149
- state definitions 141
- Application agent
 - agent functions 152
 - attributes 154
 - description 151
 - high availability fire drill 151
 - resource type definition 158
 - sample configurations 158
 - state definitions 153
- association dimension 21
- attribute data types 20
- attributes
 - Application agent 154
 - Base and Mpathd modes 87
 - Base mode 89
 - DiskGroup agent 26
 - DiskGroupSnap agent 32
 - DNS agent 101
 - ElifNone agent 201
 - FileNone agent 203
 - FileOnOff agent 205
 - FileOnOnly agent 207
 - IP agent 60
 - IPMultiNIC agent 69
 - IPMultiNICB agent 82
 - LDom agent 172
 - Mount agent 43
 - MultiNICA agent 73
 - MultiNICB agent 87
 - Multipathing mode 92
 - NFS agent 112
 - NFSRestart agent 117
 - NIC agent 66
 - NotifierMngr agent 177
 - Phantom agent 190
 - ProcessOnOnly agent 165
 - Proxy agent 187
 - RemoteGroup agent 194
 - SambaServer agent 129
 - Share agent 124
 - Volume agent 39
 - VRTSWebApp agent 184
 - Zpool agent 51
- attributes, modifying 19, 20

B

- Base mode 93
- boolean data types 20
- bundled agents 19

C

- Checklist to ensure the proper operation of
 - MultiNICB 79
- Cluster Manager (Java Console), modifying
 - attributes 20
- CNAME record 106
- configuration files
 - main.cf 191
 - modifying 20
 - types.cf 19
- configuring, Samba agents 127

D

- data type
 - boolean 20
 - string 20
- data types
 - integer 20
- description, resources 19
- dimensions
 - keylist 21
 - scalar 21
 - vector 21
- DiskGroup agent
 - agent functions 24
 - attributes 26
 - description 24
 - high availability fire drill 29
 - resource type definition 28
 - sample configurations 30
 - state definitions 26
- DiskGroupSnap agent
 - agent functions 32
 - attributes 32
 - description 31
 - resource type definition 35
 - sample configurations 35
 - state definitions 32
- DNS agent 100
 - agent functions 99

- attributes 101
- description 98
- resource type definition 105
- sample web server configuration 106

E

ElifNone agent

- agent functions 200
- attributes 201
- description 200
- resource type definition 201
- sample configuration 201

F

Fiber Channel adapter 30

FileNone agent

- agent functions 202
- attribute 203
- description 202
- resource type definition 203
- sample configurations 203

FileOnOff agent

- agent functions 204
- attribute 205
- description 204

FileOnOnly agent

- agent functions 206
- attribute 207
- description 206
- resource type definition 207
- sample configuration 207

H

high availability fire drill 29, 47, 58, 64, 105, 118, 151, 160

I

integer data types 20

Interface configuration 96

IP agent

- agent functions 59
- attributes 60
- description 58
- high availability fire drill 58
- resource type definitions 62
- sample configurations 63
- state definitions 59

IPMultiNIC agent

- agent functions 68
- attributes 69
- description 68
- resource type definitions 71
- sample configuration 71
- state definitions 69

IPMultiNICB agent 84

- agent functions 81
- description 80
- manually migrating IP address 84
- requirements 80
- resource type definition 84
- Solaris attributes 82
- state definitions 81

K

keylist dimension 21

L

LDom agent

- agent functions 171
- attributes 172
- description 170
- resource type definition 173
- sample configurations 173
- state definitions 171

M

main.cf 19, 191

modifying

- configuration files 20

modifying agents 20

monitor scenarios, DNS agent 106

Mount agent

- agent functions 40, 42
- attributes 43
- description 40
- high availability fire drill 47, 105, 118
- notes 47
- resource type definition 46
- sample configurations 48

MultiNICA agent

- agent functions 72
- attributes 73
- description 72
- resource type attributes 76

- sample configurations 77
- Solaris state definitions 73
- state definitions 73

MultiNICB agent

- administrative IP addresses 96
- agent functions 86
- attributes 87
- base and multipath, Solaris 93
- description 85
- resource type definition 93
- sample configurations 96
- state definitions 86
- test IP addresses 96

Multipathing mode 95

N

NetBIOS agent

- agent functions 135
- description 134
- resource type definition 136
- sample configurations 137
- state definitions 135

NFS agent

- agent functions 111
- attributes 112
- description 110
- resource type definition 112
- sample configurations 114
- state definitions 111

NFSRestart agent

- agent functions 116
- attributes 117
- description 115
- resource type definition 117
- sample configuration 121
- state definitions 116

NIC agent

- agent functions 65
- attributes 66
- description 64
- high availability fire drill 64
- resource type definitions 67
- sample configurations 67
- state definitions 65

NotifierMngr agent

- agent functions 176
- attributes 177
- description 176
- resource type definition 180

- sample configurations 181
- state definitions 176

O

- online query 106

P

Phantom agent

- agent functions 190
- attributes 190
- description 190
- resource type definition 190
- sample configurations 191

prerequisites

- Samba agents 126

Process agent 162

- agent functions 161
- attributes 162
- description 160
- high availability fire drill 160
- resource type definition 163
- sample configurations 163
- state definitions 161

ProcessOnOnly agent

- agent functions 164
- attributes 165
- description 164
- resource type definition 166
- sample configurations 166
- state definitions 164

Proxy agent

- agent functions 186
- attributes 187
- description 186
- resource type definition 188
- sample configurations 188

R

RemoteGroup agent

- agent functions 193
- attributes 194
- description 192
- resource type definition 198
- state definitions 193

- resource type definition 39

- SambaShare agent 133

- resource type definitions
 - Application agent 158
 - DiskGroup agent 28
 - DiskGroupSnap agent 35
 - DNS agent 105
 - ElifNone agent 201
 - FileNone agent 203
 - FileOnOnly agent 207
 - IP agent 62
 - IPMultiNIC agent 71
 - IPMultiNICB agent 84
 - LDom agent 173
 - Mount agent 46
 - MultiNICA agent 76
 - MultiNICB agent 93
 - NetBIOS agent 136
 - NFS agent 112
 - NFSRestart agent 117
 - NIC agent 67
 - NotifierMngr agent 180
 - Phantom agent 190
 - Process agent 163
 - ProcessOnOnly agent 166
 - Proxy agent 188
 - RemoteGroup agent 198
 - SambaServer agent 130
 - Share agent 124
 - Volume agent 39
 - VRTSWebApp agent 184
 - Zone agent 168
 - Zpool agent 53

- resource types 19

- resources
 - description of 19

S

- Samba agents 126
 - overview 126
 - prerequisites 126
- Samba agents configuring 127
- SambaServer agent
 - agent functions 128
 - attributes 129
 - description 128
 - resource type definition 130
 - sample configuration 130
 - state definitions 129

- SambaShare agent 131
 - agent functions 131
 - attributes 132
 - resource type definition 133
 - sample configurations 133
 - state definitions 132
- sample configurations 84
 - Apache Web server agent 149
 - Application agent 158
 - DiskGroup agent 30
 - DiskGroupSnap agent 35
 - ElifNone agent 201
 - FileNone agent 203
 - FileOnOff agent 205
 - FileOnOnly agent 207
 - IP agent 63
 - IPMultiNIC 71
 - IPMultiNICB agent 84
 - LDom agent 173
 - Mount agent 48
 - MultiNICA agent 77
 - MultiNICB agent 96
 - NetBIOS agent 137
 - NFS agent 114
 - NFSRestart agent 121
 - NIC agent 67
 - NotifierMngr agent 181
 - Phantom agent 191
 - Process agent 163
 - ProcessOnOnly agent 166
 - Proxy agent 188
 - SambaServer agent 130
 - SambaShare agent 133
 - Share agent 125
 - Volume agent 39
 - VRTSWebApp agent 185
 - Zone agent 169
 - Zpool agent 53
- scalar dimension 21
- secure DNS update 106
- Share agent
 - agent functions 123
 - attributes 124
 - description 123
 - resource type definitions 124
 - sample configurations 125
 - state definitions 124

- state definitions 100
 - Apache Web server agent 141
 - Application agent 153
 - DiskGroup agent 26
 - DiskGroupSnap agent 32
 - DNS agent 100
 - IP agent 59
 - IPMultiNIC agent 69
 - IPMultiNICB agent 81
 - LDom agent 171
 - Mount agent 42
 - MultiNICA agent 73
 - MultiNICB agent 86
 - NetBIOS agent 135
 - NFS agent 111
 - NFSRestart agent 116
 - NIC agent 65
 - NotifierMngr agent 176
 - Process agent 161
 - ProcessOnOnly agent 164
 - RemoteGroup agent 193
 - SambaServer agent 129
 - SambaShare agent 132
 - Share agent 124
 - Volume agent 39
 - VRTSWebApp agent 183
 - Zpool agent 51
- string data type 20

T

- trigger script 95
- trunking 57
- types.cf 19

V

- VCS, resource types 19
- vector dimension 21
- Volume agent
 - agent functions 38
 - attributes 39
 - description 38
 - sample configurations 39
 - state definitions 39
- volume sets, Solaris 29
- VRTSWebApp agent
 - agent functions 183
 - attributes 184
 - description 183

- resource type definition 184
- sample configuration 185
- state definitions 183

Z

- Zone agent
 - agent functions 167
 - attributes 168
 - description 167
 - resource type definition 168
 - sample configuration 169
- Zpool agent
 - agent functions 50
 - attributes 51
 - description 49
 - resource type definition 53
 - sample configurations 53
 - state definitions 51