VERITAS™

VERITAS Cluster Server 4.1

Bundled Agents Reference Guide

Solaris

N15369F

March 2005

Disclaimer

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

VERITAS Legal Notice

VERITAS Software Corporation
350 Ellis Street
Mountain View, CA 94043
USA
Phone 650–527–8000 Fax 650–527–2901
www.veritas.com

Third-Party Legal Notices

Apache Software

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work.

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

Data Encryption Standard (DES)

Support for data encryption in VCS is based on the MIT Data Encryption Standard (DES) under the following copyright:

Copyright © 1990 Dennis Ferguson. All rights reserved.

Commercial use is permitted only if products that are derived from or include this software are made available for purchase and/or use in Canada. Otherwise, redistribution and use in source and binary forms are permitted.

Copyright 1985, 1986, 1987, 1988, 1990 by the Massachusetts Institute of Technology. All rights reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided as is without express or implied warranty.

Sun Microsystems Trademarks

Sun, Solaris, SunOS, Java, Sun Java System Cluster, Sun StorEdge, Solstice DiskSuite, Sun Fire, Sun Enterprise, Online: Backup, and Netra are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

SNMP Software

SNMP support in VCS is based on CMU SNMP v2 under the following copyright:

Copyright 1989, 1991, 1992 by Carnegie Mellon University

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of CMU not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

CMU DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL CMU BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Contents

Preface

This guide provides reference information for the VCS agents bundled with VERITAS Cluster Server (VCS) software on the Solaris operating system. The guide provides information on configuring and using bundled agents.

Note that this manual does *not* cover VCS Enterprise Agents. You can find more information about VCS Enterprise Agents by referring to the *VCS Release Notes*.

How This Guide Is Organized

Chapter 1, "Introduction" on page 1, presents an overview of the agents and a description of attributes and resources.

Chapter 2, "Network Agents" on page 5, presents the network agents, such as the NIC and IP agents.

Chapter 3, "Storage Agents" on page 37, presents storage agents, such as the Mount and Volume agents.

Chapter 4, "File System Agents" on page 53, presents Network File System (NFS) agent, the NFSLock agent, and the Share agent.

Chapter 5, "Services and Applications Agents" on page 63, presents the Application, Process, and ProcessOnOnly agents. It describes the agents that make generic services and other applications highly available.

Chapter 6, "Infrastructure and Support Agents" on page 77, presents agents, such as the CampusCluster and Zone agents. It describes agents that provide high-availability for VCS-related operations.

Conventions

Convention	Usage	Example
`monospace`	Used for path names, commands, output, directory and file names, functions, and parameters.	Read tunables from the `/etc/vx/tunefstab` file. See the `ls`(1) manual page for more information.
`monospace` (**`bold`**)	Indicates user input.	# **`ls pubs`** C:\> **`dir pubs`**
italic	Identifies book titles, new terms, emphasized text, and variables replaced with a name or value.	See the *User's Guide* for details. The variable *system_name* indicates the system on which to enter the command.
bold	Depicts GUI objects, such as fields, list boxes, menu selections, etc. Also depicts GUI commands.	Enter your password in the **Password** field. Press **Return.**
blue text	Indicates hypertext links.	See "Getting Help" on page xvii.
#	Unix superuser prompt (all shells).	# **`cp /pubs/4.0/user_book /release_mgnt/4.0/archive`**

Getting Help

For technical assistance, visit http://support.veritas.com and select phone or email support. This site also provides access to resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and the VERITAS customer email notification service. Use the Knowledge Base Search feature to access additional product information, including current and past releases of product documentation.

Diagnostic tools are also available to assist in troubleshooting problems associated with the product. These tools are available on disc or can be downloaded from the VERITAS FTP site. See the `README.VRTSspt` file in the `/support` directory for details.

For license information, software updates and sales contacts, visit https://my.veritas.com/productcenter/ContactVeritas.jsp. For information on purchasing product documentation, visit http://webstore.veritas.com.

Documentation Feedback

Your feedback on product documentation is important to us. Send suggestions for improvements and reports on errors or omissions to clusteringdocs@veritas.com. Include the title and part number of the document (located in the lower left corner of the title page), and chapter and section titles of the text on which you are reporting. Our goal is to ensure customer satisfaction by providing effective, quality documentation. For assistance with topics other than documentation, visit http://support.veritas.com.

Introduction 1

Bundled agents are VCS processes that manage resources of predefined resource types according to commands received from the VCS engine, HAD. You install these agents—which are a part of VCS—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 typically:

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

Note Refer to the *VERITAS Cluster Server 4.1 User's Guide* for general information on VCS agents.

Resources and Their Attributes

Resources are the key 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 monitors an IP address resource. The agent uses the `"Address"` attribute to determine the IP address to monitor.

Modifying Agents and Their Resources

Use Cluster Manager (Java Console), Cluster Manager (Web Console), or the command line to dynamically modify the configuration of the resources managed by an agent. See the *VERITAS Cluster Server 4.1 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

Configure VCS resources with attributes. Attributes contain data about the cluster, systems, service groups, resources, resource types, and the agent. An attribute has a definition and a value. Some attributes also have default values.

Attribute Data Types

Data Type	Description
string	A string is a sequence of characters. As a best practice, enclose all strings in double quotes ("). Note that you do not have to enclose strings in quotes when they begin with a letter, and contains only: letters, numbers, dashes (-), and underscores (_). For example: ◆ A string defining a network interface such `ashme0` does not require quotes as it contains only letters and numbers. Enclosing the string in double quotes is also acceptable—`"ashme0"`. ◆ A string defining an IP address requires quotes: `"192.168.100.1"` because the address contains periods. A string can contain double quotes, but the quotes must be immediately preceded by a backslash. In a string, represent a backslash with two forward 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.
boolean	A boolean is an integer with the possible values of `0` (false) and `1` (true).

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 in that list.
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{}`.

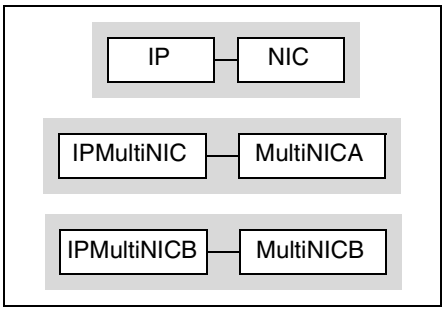
Network Agents 2

Network agents make IP addresses highly available.

- The "IP Agent" on page 8 and the "NIC Agent" on page 11 work together to make a virtual IP address highly available.
- The "MultiNICA Agent" on page 17 and the "IPMultiNIC Agent" on page 14 work together to make a virtual IP address, configured on servers with multiple adapters, highly available.
- The "MultiNICB Agent" on page 27 and the "IPMultiNICB Agent" on page 23 work together to make a virtual IP address, configured on servers with multiple adapters, highly available. With this combination, you can use base or multipathing modes.

Overview

These agents always work together in pairs: IP and NIC, IPMultiNIC and MultiNICA, and IPMultiNICB and MultiNICB.



IP and NIC Agents

- Monitor a single NIC

IPMultiNIC and MultiNICA Agents

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

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
- Cannot transfer the original base IP address
- Provide faster failover compared to MultiNICA but requires more IP addresses
- Have more than one active NIC at a time
- Supports IP multipathing and trunking

Defining IP Addresses

Here are some of terms used to describe IP addresses this guide:

Logical—any IP address assigned to a NIC.

Administrative—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.

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

Floating and virtual—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.

Test—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.

IP Agent

The IP agent assigns a virtual IP address to the network interface card (NIC), monitors the IP address, and removes it. The agent also monitors the associated subnet mask on a NIC. You must plumb the interface with the base IP address before you configure the IP agent. The virtual IP address specified in the configuration must not be one currently in use.

Entry Points

- Online—Plumbs the IP address to the NIC. Checks if another system is using the IP address. Uses the `ifconfig` command to set the IP address on a unique alias on the interface.
- Offline—Brings down the IP address associated with the specified interface.
- Monitor—Monitors the interface to test if the IP address associated with the interface is alive.
- Clean—Brings down the IP address associated with the specified interface.

Type Definition

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

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
Address	A virtual IP address, which is different from the base IP address, and which is associated with the interface. ◆ Type and dimension: string-scalar ◆ Example: `"192.203.47.61"`
Device	The name of the NIC device associated with the IP address. Contains the device name without an alias. ◆ Type and dimension: string-scalar ◆ Example: `"le0"`

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
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`
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 netmask associated with the IP address of the resource. Specify the value of the netmask in decimal (base 10) or hexadecimal (base 16). **Note** VERITAS 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 based on the contents of the `/etc/netmasks` for a given address range. ◆ Example: `"255.255.248.0"` ◆ Default: `"255.0.0.0"` If the `ifconfig` command is executed without a netmask argument.

Options	Options for the `ifconfig` command. ◆ Type and dimension: string-scalar ◆ Default: n/a ◆ Example: "`trailers`"

Sample Configurations

Sample 1

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

Sample 2: NetMask in decimal (base 10)

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

Sample 3: NetMask in hexadecimal (base 16)

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

NIC Agent

Monitors the configured NIC. If a network link fails, or if a problem arises with the device card, the resource is marked FAULTED. The NIC listed in the Device attribute must have an administrative IP address, which is the default IP address assigned to the physical interface of a host on a network. This agent does not configure network routes or administrative IP addresses.

Before using 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. (If necessary, refer to the NIC documentation.)

Entry Point

- 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 FAULTED.

State Definitions

- ONLINE—Indicates that the NIC is working.
- FAULTED—Indicates that the NIC has failed.
- UNKNOWN—Indicates that the device is not configured correctly.

Type Definition

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


Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
Device	Name of the NIC. ◆ Type and dimension: string-scalar ◆ Example: "le0"

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
NetworkHosts	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 HostName, 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 currently only supports Ethernet (ether). ◆ Type and dimension: string-scalar ◆ Default: "ether"
PingOptimize	Number of monitor cycles to detect if configured interface is inactive. 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

Sample Configurations

Without Network Hosts (Using Default Ping Mechanism)

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

With Network Hosts

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


IPMultiNIC Agent

Works with the MultiNICA agent. Manages the virtual IP address 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 should have the MultiNICA resource. The other groups have Proxy resources pointing to it.

Entry Points

- Online—Configures a virtual IP address on one interface of the MultiNICA resource.
- Offline—Removes a 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.

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 resource is configured inaccurately in the `main.cf` file.

Type Definition

```
type IPMultiNIC (
        static str ArgList[] = { "MultiNICResName:Device", Address,
        NetMask, "MultiNICResName:ArpDelay", Options,
        "MultiNICResName:Probed", MultiNICResName, IfconfigTwice }
        static int MonitorTimeout = 120
        str Address
        str NetMask
        str Options
        str MultiNICResName
        int IfconfigTwice = 0
)
```

Required Attributes

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

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
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 netmask associated with the IP address of the resource. Specify the value of the netmask in decimal (base 10) or hexadecimal (base 16). **Note** VERITAS 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 based on the contents of the `/etc/netmasks` for a given address range. ◆ Example: "255.255.248.0"
Options	Options for the `ifconfig` command. ◆ Type and dimension: string-scalar ◆ Example: "trailers"

Note VERITAS recommends that you set the RestartLimit for IPMultiNIC resources to a greater-than-zero value. This 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.

Sample Configuration: IPMultiNIC and MultiNICA

For details on the following example, refer to "Sample Configuration: MultiNICA and IPMultiNIC" on page 21.

```
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

Works with the IPMultiNIC agent. Represents a set of network interfaces and provides failover capabilities between 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:

1. A MultiNICA resource in one of the service groups.

2. Proxy resources that point to the MultiNICA resource in the other service groups.

Entry Point

- Monitor—Checks for activity on a configured interface by sampling input packets received on that interface. If it does not detect activity, it forces activity by sending out a broadcast ping. If it detects a failure, it migrates to the next available interface configured in the Device attribute.

Type Definition

```
type MultiNICA (
        static str ArgList[] = { Device, NetMask, ArpDelay,
        RetestInterval, Options, RouteOptions, PingOptimize,
        MonitorOnly, IfconfigTwice, HandshakeInterval, NetworkHosts}
        static int MonitorTimeout = 300
        static int OfflineMonitorInterval = 60
        static str Operations = None
        str Device{}
        str NetMask
        int ArpDelay = 1
        int RetestInterval = 5
        str Options
        str RouteOptions
        int PingOptimize = 1
        int IfconfigTwice = 0
        int HandshakeInterval = 20
        str NetworkHosts[]
)
```


Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
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" }`

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
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`
HandshakeInterval	Computes the maximum number of attempts 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 attempt may take about 10 seconds. To prevent spurious failovers, the agent must try to contact a host on the network several times before marking 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 is the equivalent to two attempts (20/10).
IfconfigTwice	Causes an IP address to be configured twice, using an `ifconfig up-down-up` sequence. Increases the probability of gratuitous ARP requests (caused by `ifconfig up`) to reach clients. ◆ Type and dimension: integer-scalar ◆ Default: `0`

NetMask	Netmask for the base IP address. You can specify the value of NetMask in decimal (base 10) or hexadecimal (base 16). **Note** VERITAS recommends that you specify a netmask for each virtual interface. ◆ Type and dimension: string-scalar ◆ Default: + Example: `"255.255.255.0"`
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 HostName) to prevent the monitor from timing out (DNS causes the ping to hang). If this attribute is not specified, 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: `{ "166.93.2.1", "166.97.1.2" }`
Options	The `ifconfig` 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. **Note** 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"`

MultiNICA Notes

- If all NICs configured in the Device attribute are down, the MultiNICA agent faults the resource after a two to three minute interval. This delay occurs because the MultiNICA agent tests the failed NIC several times before marking the resource OFFLINE. Messages recorded in the engine log during failover provide a detailed description of the events that take place. (The engine log is located 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.

  For example, you have two active NICs, `hme0` (`10.128.2.5`) and `qfe0` (`10.128.2.8`), and 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, making 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 Configuration: MultiNICA and IPMultiNIC

In the following example, two machines, sysa and sysb, each have a pair of network interfaces, `le0` and `qfe3`. The two interfaces, `le0` and `qfe3`, have the same base, or physical, IP address. However, the addresses on different hosts can differ. 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 logical IP addresses are configured by the IPMultiNIC agent. The resources ip1 and ip2, shown in the following example, have the Address attribute which 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 to point to the MultiNICA resource in the first group. This prevents redundant monitoring of the NICs on the same system. The IPMultiNIC resource is always made dependent on the MultiNICA resource. See "IPMultiNIC Agent" on page 14.

```
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
```

IPMultiNICB Agent

Works with the MultiNICB agent. Configures and manages virtual IP addresses (IP aliases) on an active network device specified by the MultiNICB resource. When the MultiNICB agent reports a particular interface as failed, the IPMultiNICB agent moves the IP address to the next active interface.

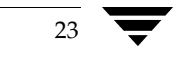
If multiple service groups have IPMultiNICB resources associated with the same MultiNICB resource, only one group should have a MultiNICB resource. The other groups should have a proxy resource pointing to the MultiNICB resource.

Entry Points

- Online—Finds a working interface with the appropriate interface alias or interface name, and configures the logical IP address on it. Erases previous failover information created by the MultiNICB resource for this logical IP address.
- 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 no working interfaces are available, or the current interface fails, monitor returns OFFLINE.

State Definitions

- ONLINE—Indicates that the IP address specified in the Address attribute is up on one of the working network interfaces of the resource specified in the BaseResName attribute.
- OFFLINE—Indicates that the IP address specified in the Address attribute is not up on any of the working network interfaces of the resource specified in the BaseResName attribute.
- UNKNOWN—Indicates that the status of the network interfaces is unknown.



Type Definition

```
type IPMultiNICB (
        static str ArgList[] = {BaseResName, Address, NetMask,
        DeviceChoice}
        str BaseResName
        str Address
        str NetMask
        str DeviceChoice = "0"
)
```

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
Address	The logical IP address that the IPMultiNICB resource must handle. ◆ Type and dimension: string-scalar ◆ Example: `"10.128.10.15"`
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. ◆ Type and dimension: string-scalar ◆ Example: "`gnic_n`"
NetMask	Netmask associated with the logical IP address. ◆ Type and dimension: string-scalar ◆ Example: `"255.255.255.0"`

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example

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"`
	◆ Example:
	`DeviceChoice = "qfe0"`
	◆ Example:
	`DeviceChoice = "1"`

NetMask	Netmask for the base IP address. You can specify the value of NetMask in decimal (base 10) or hexadecimal (base 16). **Note** VERITAS recommends that you specify a netmask for each virtual interface. ◆ Type and dimension: string-scalar ◆ Default: + Example: `"255.255.255.0"`

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 VCS IP agent (IPMultiNICB, IPMultiNIC, or IP) can control each logical IP address.

Sample Configuration: IPMultiNICB and MultiNICB

Refer to "VCS Configuration" on page 36 for a sample configuration of IPMultiNICB and MultiNICB.

Manually Migrating a Logical IP Address

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

Usage:

```
# haipswitch -s MultiNICB resname
# haipswitch MultiNICB resname IPMultiNICB resname ip addr netmask
  from to
```

In the first form, the command shows the status of the interfaces for the specified MultiNICB resource. In the second form, the command uses the following steps:

1. Checks that both *from* and *to* interfaces are associated with the specified MultiNICB resource and the *to* interface is working. If not, the command aborts the operation.
2. Removes the IP address on the *from* logical interface.
3. Configures the IP address on the *to* logical interface.
4. Erases previous failover information created by MultiNICB for this logical IP address.

MultiNICB Agent

Works with the IPMultiNICB agent. Allows IP addresses to fail over to multiple NICs on the same system, before VCS attempts to fail over to another system.

When you use the MultiNICB agent, you must plumb the NIC before putting it under the agent's control. You must configure all the NICs on a single IP subnet inside a single MultiNICB resource.

Modes of Operation

MultiNICB has two modes of operation depending on the UseMpathd attribute: "Base Mode" and "Multipathing Mode."

Base Mode

The agent monitors the interfaces it controls by sending packets to other hosts on the network and checking the link status of the interfaces.

If a NIC goes down, the MultiNICB agent notifies the IPMultiNICB agent, which then 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 agent uses as the test IP address.

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.

In this mode, MultiNICB uses the following criteria to determine if an interface is working:

- Interface status: The interface status as reported by driver of the interface (assuming the driver supports this feature). This test is skipped if the attribute `IgnoreLinkStatus = 1`.
- ICMP echo: ICMP echo request packets are sent to one of the network hosts (if specified). Otherwise, the agent uses ICMP broadcast and caches the sender of the first reply as a network host. While sending and receiving ICMP packets, the IP layer is completely bypassed.

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 Failback

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.

This mode's default is UseMpathd set to `0`.

This mode works when you set UseMpathd to `0`.

Multipathing Mode

You can configure the MultiNICB agent to work with the IP multipathing daemon. The MultiNICB agent relies on the IP Multipathing daemon (see the man page: `in.mpathd (1M)`) to detect network failures and repairs. In this situation, MultiNICB limits its functionality to monitoring the FAILED flag on physical interfaces and monitoring the mpathd process.

This mode only works when you set UseMpathd to `1`.

Entry Points

- Open—Allocates an internal structure to store information about the resource.
- Close—Frees the internal structure 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 failover. Performs failback if failback is set to 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.
- OFFLINE—Indicates that all of the network interfaces listed in the Device attribute failed.
- UNKNOWN—Indicates that the network interfaces are not configured accurately.

Type Definition

```
type MultiNICB (
        static int MonitorInterval = 10
        static int OfflineMonitorInterval = 60
        static int MonitorTimeout = 60
        static int Operations = None

        static str ArgList[] = { UseMpathd, MpathdCommand,
        ConfigCheck, MpathdRestart, Device, NetworkHosts,
        LinkTestRatio, IgnoreLinkStatus, NetworkTimeout,
        OnlineTestRepeatCount, OfflineTestRepeatCount, NoBroadcast,
         DefaultRouter, Failback}

        int UseMpathd = 0
        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 = 0

        str DefaultRouter = "0.0.0.0"

        int Failback = 0
)
```


Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example

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
	◆ Example:
	`Device = { "qfe0" , "qfe4" }`
	◆ Example:
	`Device = { "qfe0" = 0, "qfe1" = 2, "qfe2" = 3 }`
	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.

Optional Attributes for Base and Mpathd Modes

Optional Attribute	Description, Type and Dimension, Default, and Example
MpathdCommand	This 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: `"/sbin/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. If set to `0`, in.mpathd is automatically killed on that system. For more information about mpathd, refer to the Sun documentation. If set to `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

Optional Attribute	Description, Type and Dimension, Default, and Example
DefaultRouter	This is the IP address of the default router on the subnet. If specified, 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; otherwise, 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"`
Failback	If set to `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 set to `1`, the agent ignores the driver-reported interface status while testing the interfaces. If set to `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 using trunked interfaces (for example, Sun Trunking), you must set this attribute to `1`. Otherwise set it to `0`. ◆ Type and dimension: integer-scalar ◆ Default: `1`

LinkTestRatio	This 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 set to `1`, packets are sent during every monitor cycle. If set to `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 entry points.
NetworkHosts	List of host IP addresses on the IP subnet that are pinged to determine if the interfaces are working. 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" }`
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 set to 1, NoBroadcast prevents MultiNICB from sending broadcast ICMP packets. (Note: MultiNICB can still send ARP requests.) If NetworkHosts are not specified and NoBroadcast is set to `1`, the MultiNICB agent cannot function properly. **Note** VERITAS 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
OnlineTestRepeatCount	Number of times the test is repeated if the interface status changes from DOWN to UP. This helps to avoid oscillations in the status of the interface. ◆ Type and dimension: integer-scalar ◆ Default: 3

Optional Attributes for Multipathing Mode

Optional Attribute	Description, Type and Dimension, Default, and Example
ConfigCheck	If set to 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 set to 1, MultiNICB attempts to restart mpathd. Valid values for this attribute are 0 and 1. ◆ Type and dimension: integer-scalar ◆ Default: 1

Checklist for Using 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 plumb all the interfaces that are under the MultiNICB resource and given them test IP addresses.
- ✔ All test IP addresses for the MultiNICB resource must belong to the same subnet as the virtual IP address.

Tip The base IP addresses, which the agent uses to test the link status, should be reserved 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 NetworkHosts is specified, the hosts must be directly accessible on the LAN.
- ✔ 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`.

Trigger Script

MultiNICB monitor entry point calls a VCS trigger in case of an interface going up or down. The following arguments are passed to the script:

- MultiNICB resource name
- device whose status changed (for example, `qfe0`)
- device's previous status (`0` for down, `1` for up)
- 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 clusterresource improved" traps which are mentioned in the *VCS User's Guide*. A sample `mnicb_postchange` trigger is provided with the agent. The user may 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 has 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 eprom variable to complete the address setup. The base IP addresses must be configured on the interfaces before the MultiNICB agent controls the interfaces. This can be completed at system start up using `/etc/hostname.`*`XXX`* initialization files as in the examples below.

Setting Up Test and Administrative IP Addresses

These examples demonstrate setting up test and administrative IP addresses for your clustered systems. You do *not* need to perform the following steps for the floating IP addresses, as the agent takes care of this automatically. See "Defining IP Addresses" on page 7.

Sample of Setting Up Test and Administrative IP Addresses

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

VCS Configuration

The following is an example VCS 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 (
)
```

Storage Agents 3

This chapter contains the following agents:

Disk Agent

Manages a raw disk.

Entry Point

- Monitor—Determines if the disk is accessible by performing read I/O operations on raw disk.

Type Definition

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

Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
Partition	Indicates which partition to monitor. Specify the partition with the full path beginning with a slash (/). If this path is not specified, the name is assumed to reside in `/dev/rdsk`. ◆ Type and dimension: string-scalar ◆ Example: `"/dev/rdsk"`

Sample Configuration

```
Disk c1t0d0s0 (
  Partition = c1t0d0s0
  )
```

DiskGroup Agent

Brings online, takes offline, and monitors a VERITAS Volume Manager (VxVM) disk group. This agent uses VxVM commands.

Entry Points

- 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 `vxdg` command. If the disk group was imported with `noautoimport=off`, then the DiskGroup agent changes the value of `noautoimport=on` instead of taking the service group offline. The VxVM in 4.0 MP1 provides this option to change the auotimport attribute.
- Clean—Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.
- Info—The DiskGroup info entry point gets information from the volume manager and displays the type and free size for the DiskGroup resource.

State Definitions

- ONLINE—Indicates that the disk group is imported.
- OFFLINE—Indicates that the disk group is not imported.
- UNKNOWN—Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

Type Definition

```
type DiskGroup (
        static int OnlineRetryLimit = 1
        static str ArgList[] = { DiskGroup, StartVolumes, StopVolumes,
        MonitorOnly, MonitorReservation, tempUseFence }
        str DiskGroup
        str StartVolumes = 1
        str StopVolumes = 1
        static int NumThreads = 1
        boolean MonitorReservation = 0
        temp str tempUseFence = "INVALID"
)
```


Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
DiskGroup	Name of the disk group configured with VERITAS Volume Manager. ◆ Type and dimension: string-scalar ◆ Example: `"diskgroup1"`

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
StartVolumes	If value is `1`, the DiskGroup `online` script 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` script stops all volumes belonging to that disk group before deporting the group. ◆ Type and dimension: string-scalar ◆ Default: `"1"`
MonitorReservation	If value set to `1`, the agent monitors the SCSI reservation on the disk group. If reservation is missing, it takes the resource offline. ◆ Type and dimension: boolean-scalar ◆ Default: `0`
TempUseFence	Do not use. For VERITAS use only. ◆ Type and dimension: string-scalar

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 enables VCS to control the importation and deportation of disk groups as needed when bringing disk groups online and taking them offline.

Note If you enable a disk group configured as a DiskGroup resource that does *not* have the `noautoimport` flag set to true, VCS changes the `noautoimport` flag to true. VxVM provides this new option from version 4.0 MP1.

To check the status of the `noautoimport` flag for an imported disk group, type:

```
# vxprint -l disk_group | grep noautoimport
```

The following command changes the autoimport flag to false:

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

Info Entry Point

The following steps are necessary to initiate the info entry point by setting the InfoInterval timing to a value greater than 0. For example,

```
# haconf -makerw
# hatype -modify DiskGroup InfoInterval 60
```

In this case, the info entry point will get executed every 60 seconds. The command to retrieve information about the DiskType and FreeSize of the DiskGroup resource is:

```
# hares -value diskgroupres ResourceInfo
```

Output will include the following information:

```
DiskType sliced
FreeSize 35354136
```

Sample Configuration

```
DiskGroup dg1 (
  DiskGroup = testdg_1
  )
```


Volume Agent

Brings online, takes offline, and monitors a VERITAS Volume Manager (VxVM) volume.

Entry Points

- Online—Starts the volume using the `vxrecover` command.
- Offline—Stops the volume using the `vxvol` 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.

Type Definition

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

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
DiskGroup	Name of the disk group that contains the volume. ◆ Type and dimension: string-scalar ◆ Example: `"sharedg"`
Volume	Name of the volume. ◆ Type and dimension: string-scalar ◆ Example: `"vol3"`

Sample Configuration

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


DiskReservation Agent

Reserves and monitors SCSI disks for a system, enabling a resource to go online on that system. The agent supports all SCSI II disks.

Use this agent to specify a list of raw disk devices, and reserve all or a percentage of accessible disks for an application. The reservations prevent disk data corruption by restricting other systems from accessing and writing to the disks.

An automatic probing feature allows systems to maintain reservations even when the disks or bus are reset. The optional FailFast feature minimizes data corruption in the event of a reservation conflict by causing the system to panic.

The DiskReservation agent is supported on Solaris 2.7 and above. The agent is not supported with dynamic multipathing software, such as VERITAS DMP.

Entry Points

- Online—Brings the resource online after reserving all or a specified percentage of accessible disks.
- Offline—Releases reservations on reserved disks.
- Monitor—Monitors the accessibility and reservation status of the reserved disks.
- Clean—Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.

State Definitions

- ONLINE—Indicates that the number of reserved disks is greater than or equal to the percentage specified in the resource definition.
- OFFLINE—Unable to reserve disks.
- UNKNOWN—Indicates that a problem exists with the configuration.

Type Definition

```
type DiskReservation (
        static str ArgList[] = { Disks, FailFast, ConfigPercentage,
        ProbeInterval }
        str Disks[]
        boolean FailFast = 0
        int ConfigPercentage = 80
        int ProbeInterval = 6
)
```

Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
Disks	A list of raw disk devices. Use the absolute or relative device path. The order of the disks in the list must be the same across all systems in the cluster, even if the same device has a different name on different systems. **Note** You must change this attribute before bringing a resource online. An online device must be taken offline before altering this attribute because disk reservation occurs *during* the process of bringing a resource online. ◆ Type and dimension: string-vector ◆ Example: `{ "c1t2d0s4" }`

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
FailFast	If enabled, FailFast causes the system to panic when a reservation conflict is detected, reducing the chance of further data corruption. ◆ Type and dimension: boolean-scalar ◆ Default: `0`
ConfigPercentage	The minimum percentage of configured disks that can be reserved before the resource can go online. ◆ Type and dimension: string-scalar ◆ Default: `"80"`
ProbeInterval	Changes how many seconds the automatic probe function checks the reservation status of the disks. A lower value for ProbeInterval specifies more frequent probes and provides for quicker discovery of reservation conflicts. ◆ Type and dimension: string-scalar ◆ Default: `"6"`

Sample Configurations

Sample 1

Configuring the Mount resource to depend on the DiskReservation resource when a disk is mounted on a system ensures that no other system can mount the same disk. The group definition in the `main.cf` would resemble the following definition.

```
group groupx (
  SystemList = { sysa, sysb }
  AutoStartList = { sysa }
  )
  DiskReservation dr (
    Disks = { c1t2d0s4 }
    FailFast = 1
    )
  Mount export1 (
    MountPoint = "/test"
    BlockDevice = "/dev/dsk/c1t2d0s4"
    FSType = ufs
    MountOpt = rw
    )

    export1 requires dr

    // resource dependency tree
    //
    //      group groupx
    //      {
    //      Mount export1
    //          {
    //          DiskReservation dr
    //          }
    //      }
```

Sample 2

You can use the DiskReservation resource in conjunction with VERITAS Volume Manager. In the following example of a group defined in the `main.cf`, the Mount resource depends on the Volume resource, which depends on the DiskGroup resource, which depends on the DiskReservation resource.

The Disks list for the DiskReservation resource must contain the raw devices that constitute the disk group. Use the VxVM `vxdisk list` command to list the devices constituting the volume.

Five disks exist in the user-defined group. Four of these disks must be accessible and reserved for the resource group to go online (the default `ConfigPercentage` is `80`).

FailFast preserves data integrity by directing the system to panic if a conflict of reservations arises between systems.

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

  DiskReservation dr (
    Disks = { "/dev/rdsk/c0t2d0s2", "/dev/rdsk/c1t2d0s2",
          "/dev/rdsk/c2t2d0s2", "/dev/rdsk/c3t2d0s2",
          "/dev/rdsk/c4t2d0s2" }
    FailFast = 1
    )

  DiskGroup resdg (
    DiskGroup = resdg
    )

  Volume resvol (
    Volume = resvol
    DiskGroup = resdg
    )

  Mount export1 (
    MountPoint = "/test"
    BlockDevice = "/dev/vx/dsk/resdg/resvol"
    FSType = ufs
    MountOpt = rw
    )

  export1 requires resvol
  resvol requires resdg
  resdg requires dr
```


```
// resource dependency tree
//
//      group groupx
//      {
//      Mount export1
//          {
//          Volume resvol
//              {
//              DiskGroup resdg
//                  {
//                  DiskReservation dr
//                  }
//              }
//          }
//      }
```

Mount Agent

Brings online, takes offline, and monitors a file system mount point.

Entry Points

- Online—Mounts a block device on the directory. If the mount process fails, the agent attempts to run the `fsck` command on the raw device to remount the block device.
- Offline—Unmounts the file system.
- Monitor—Determines if the file system is mounted. Checks mount status using the `stat` and `statvfs` commands.
- Clean—Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.
- Info—See description on page 52

State Definitions

- ONLINE—Indicates that the block device is mounted on the specified mount point.
- OFFLINE—Indicates that the block device is not mounted on the specified mount point.
- UNKNOWN—Indicates that a problem exists with the configuration.

Type Definition

```
type Mount (
        static str ArgList[] = { MountPoint, BlockDevice, FSType,
MountOpt, FsckOpt, SnapUmount, CkptUmount, SecondLevelMonitor,
        SecondLevelTimeout }
        str MountPoint
        str BlockDevice
        str FSType
        str MountOpt
        str FsckOpt
        int SnapUmount = 0
        int CkptUmount = 1
    boolean SecondLevelMonitor = 0
    int SecondLevelTimeout = 30
)
```


Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
BlockDevice	Device for mount point. ◆ Type and dimension: string-scalar ◆ Example: `"/dev/vx/dsk/campus-dg1/campus-vol1"`
FsckOpt	Options for `fsck` command. You must include `-y` or `-n` must as arguments to `fsck`, otherwise the resource cannot come online. VxFS file systems perform a log replay before a full `fsck` operation (enabled by `-y`) takes place. Refer to the manual page on the `fsck` command for more information. ◆ Type and dimension: string-scalar
FSType	Type of file system. Your choices are: vxfs, ufs, or nfs. ◆ Type and dimension: string-scalar ◆ Example: `"nfs"`
MountPoint	Directory for mount point. ◆ Type and dimension: string-scalar ◆ Example: `"/campus1"`

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
MountOpt	Options for the `mount` command. To see a list of available options, refer to the `mount` command's man page. ◆ Type and dimension: string-scalar ◆ Example: `"rw"`
SnapUmount	If set to `1`, this attribute automatically unmounts VxFS snapshots when the file system is unmounted. ◆ Type and dimension: integer-scalar ◆ Default: `0`
CkptUmount	If set to `1`, this attribute automatically unmounts VxFS checkpoints when the file system is unmounted. ◆ Type and dimension: integer-scalar ◆ Default: `1`
SecondLevelMonitor	This attribute is only applicable to NFS. If set to `1`, this attribute enables detailed monitoring of a NFS mounted file system. ◆ Type and dimension: boolean-scalar ◆ Default: `0`
SecondLevelTimeout	This attribute is only applicable to NFS. This is the timeout (in seconds) for the SecondLevelMonitor attribute. ◆ Type and dimension: integer-scalar ◆ Default: `30`

Info Entry Point

The Mount info entry point executes the command:

```
df -k <mount_point>
```

The output displays Mount resource information:

```
Size Used Avail Use%
```

The following steps are necessary to initiate the info entry point by setting the InfoInterval timing to a value greater than 0. For example,

```
haconf -makerw
hatype -modify Mount InfoInterval 60
```

In this case, the info entry point will get executed every 60 seconds. The command to retrieve information about the Mount resource is:

```
hares -value mountres ResourceInfo
```

Output will include the following information:

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

Sample Configuration

```
Mount campus-fs1 (
MountPoint= "/campus1"
BlockDevice = "/dev/vx/dsk/campus-dg1/campus-vol1"
FSType = "vxfs"
FsckOpt = "-n"
MountOpt = "rw"
  )
```

File System Agents 4

This chapter contains the following agents:

NFS Agent

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

Service Management Facility—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 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 to store persistent configuration information as well as runtime data for all the services. Thus, 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 NFS daemons under VCS control**

- Disable SMF for nfsd and mountd.

```
svccfg delete -f svc:/network/nfs/server:default
```

Entry Points

- Online—Checks if nfsd and mountd processes are running. If they are not running, the agent starts the processes and exits.
- Monitor—Monitors versions 2 and 3 of the nfsd process, and versions 1, 2, and 3 of the mountd process. Monitors TCP and UDP versions of the processes by sending RPC (Remote Procedure Call) calls `clnt_create` and `clnt_call` to the RPC server. If the calls succeed, the resource is reported ONLINE.
- Clean—Kills and restarts the nfsd and mountd processes.

State Definitions

- ONLINE—Indicates that the NFS daemons are running properly.
- OFFLINE—Indicates that the NFS daemons are not running properly.
- FAULTED—Indicates that the NFS daemons are not running properly.
- UNKNOWN—Unable to determine the status of the NFS daemons.

Type Definition

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

        str Protocol = all
```

Optional Attribute

Optional Attribute	Description, Type and Dimension, Default, and Example
Nservers	Specifies the number of concurrent NFS requests the server can handle. ◆ Type and dimension: integer-scalar ◆ Default: 16 ◆ Example: 24

Sample Configuration

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


NFSLock Agent

The NFS record lock recovery agent (NFSLock agent) recovers NFS record locks after sudden reboots or crashes on clients and servers. This avoids file corruption and provides the high availability of NFS record locks.

The NFSLock agent brings online, takes offline, and monitors the three daemons: smsyncd, statd, and lockd. Some limitations apply to the NFSLock agent, see "Limitations" on page 58.

Service Management Facility—Solaris 10

You must disable the Service Management Facility (SMF) for NFS daemons for the NFSLock 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 NFSLock agent online or the agents returns an UNKNOWN state.

▼ To keep NFS daemons under VCS control

1. Disable SMF for statd.

   ```
   # svccfg delete -f svc:/network/nfs/status:default
   ```

2. Disable SMF for lockd.

   ```
   # svccfg delete -f svc:/network/nfs/nlockmgr:default
   ```

Execution of above commands stops lockd, statd, and automountd daemons running on the system. Therefore, you need to restart lockd, statd, and automountd manually after executing the commands.

▼ **To manually restart the 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**

Entry Point

- Online—Kills lockd and statd.
  - Kills lockd to start the grace period that allows all clients holding locks, at time of the crash, an opportunity to reclaim their locks before the server revokes it and resumes normal operation.
  - Kills statd to trigger the recovery processing.
- Offline—Kills statd to break the connection between client and server's statd and kills lockd to release the locks held.
- Monitor—Monitors lockd, statd, and smsyncd, and restarts them if they are not running.
- Clean—Kills statd to break the connection between client and server's statd and kills lockd to release the locks held.
- nfs_restart—The nfs_restart trigger runs smsyncd in oneshot mode to sync up the state on the failed-to node and then starts smsyncd, statd, and lockd.

State Definitions

- ONLINE—Indicates that the lock agent is running properly.
- OFFLINE—Indicates that the lock agent is not running properly
- UNKNOWN—Indicates inability to determine the status of the NFSLock agent.

Type Definition

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


Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
PathName	The path of the root directory of the file system for which file record locks are being managed. ◆ Type and dimension: string-scalar ◆ Example: `"/mnt/test_mnt"`

Limitations

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

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

  a. `/etc/hosts`

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

  b. `/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` 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.

- If there are multiple service groups online on each node in the cluster, then failing over a single service group disrupts the locking service for all other groups online on both the failing-from and failing-to nodes.
- 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 second, 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 Configuration

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

        IP IPres (
                Device = hme0
                Address = "10.122.12.253"
                NetMask = "255.255.240.0"
                )

        Mount Mountres (
                MountPoint = "/mnt/test_mnt"
                BlockDevice = "/dev/dsk/c1t11d0s2"
                FSType = ufs
                MountOpt = rw
                FsckOpt = "-y"
                )

        NFS NFSres (
        )

        NFSLock NFSLockres (
                PathName = "/mnt/test_mnt"
                )

        NIC NICres (
                Device = hme0
                )

        Share Shareres (
                PathName = "/mnt/test_mnt"
```


```
        Options = "-o rw"
        )

IPres requires NICres
IPres requires Shareres
NFSLockres requires Mountres
Shareres requires NFSLockres
Shareres requires NFSres


// resource dependency tree
//
//      group NFSgrp
//      {
//      IP IPres
//          {
//          NIC NICres
//          Share Shareres
//              {
//              NFSLock NFSLockres
//                  {
//                  Mount Mountres
//                  }
//              NFS NFSres
//              }
//          }
//      }
```

Share Agent

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

Entry Points

- 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.

Type Definition

```
type Share (
        static str ArgList[] = { PathName, Options }
        str PathName
        str Options
)
static int NumThreads = 1
```

Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
PathName	Pathname of the file system to be shared. ◆ Type and dimension: string-scalar ◆ Example: `"/share1x"`

Optional Attribute

Optional Attribute	Description, Type and Dimension, Default, and Example
Options	Options for the `share` command. ◆ Type and dimension: string-scalar ◆ Example: `"-o rw"`

Sample Configuration

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

Services and Applications Agents 5

This chapter contains the following agents:

Application Agent

Brings applications online, takes them offline, and monitors their status. Enables you to specify different executables for the online, offline, and monitor routines. (An application has an executable to start it and an executable to stop it.) The executables must exist locally on each node. By default, an application runs in the context of root. Specify the user name to run an application in a user context.

The agent starts and stops the application with user-specified programs.

Monitor the application in the following ways:

- Use the monitor program
- Specify a list of processes
- Specify a list of process ID files
- All or some of the above

Entry Points

- Online—Runs the StartProgram with the specified parameters in the specified user context.
- Offline—Runs the StopProgram with the specified parameters in the specified user context.
- Monitor—If you specify the MonitorProgram, the agent executes the user-defined MonitorProgram in the user-specified context. If you specify PidFiles, the routine verifies that the process ID found in each listed file is running. If you specify MonitorProcesses, the routine verifies that each listed process is running in the user-specified context.

  MonitorProgram must return ONLINE to employ any other monitoring method. Any one, two, or three of these attributes can be used to monitor the application. If any one process specified in either PidFiles or MonitorProcesses is determined not to be running, the monitor returns OFFLINE.
- Clean—Kills processes specified in PidFiles or MonitorProcesses. Ensures that only those processes (specified in MonitorProcesses) 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 specified in PidFiles and MonitorProcesses are running and that the MonitorProgram returns ONLINE.
- OFFLINE—Indicates that at least one process specified in PidFiles or MonitorProcesses is not running, or that the MonitorProgram returns OFFLINE.
- UNKNOWN—Indicates an indeterminable application state.

Type Definition

```
type Application (
    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
)
```


Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
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 are separated by spaces. For applications running in Solaris 10 zones, use the path as seen from the zone that is running the application——its relative path. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/sbin/samba start"` ◆ Solaris 10—example in global and non-global zones: `"/usr/sbin/samba start"`
StopProgram	The executable, created locally on each node, that stops the application. Specify the complete path of the executable. Applicable command-line arguments follow the name of the executable and are separated by spaces. For applications running in Solaris 10 zones, use the path as seen from the zone that is running the application——its relative path. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/sbin/samba stop"` ◆ Solaris 10—example in global and non-global zones: `"/usr/sbin/samba stop"`
At least one of the following attributes: ◆ MonitorProcesses ◆ MonitorProgram ◆ PidFiles	See "Optional Attributes" on page 67.

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
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 are separated by spaces. For applications running in Solaris 10 zones, use the path as seen from the zone that is running the application——its relative path. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/sbin/samba force stop"` ◆ Solaris 10—example in global and non-global zones: `"/usr/sbin/samba force stop"`
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"`
MonitorProcesses	A list of processes that you want monitored and cleaned. Each process name is the name of an executable. Qualify the executable name with its complete path if the path starts the executable. The process name must be the full command line argument displayed by the `ps -u <user> -o args | more` command for the process. ◆ Type and dimension: string-vector ◆ Example: `{ "nmbd" }`

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 are separated by spaces. For applications running in Solaris 10 zones, use the path as seen from the zone that is running the application——its relative path. 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. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/local/bin/sambaMonitor all"` ◆ Solaris 10—example in global and non-global zones: `"/usr/local/bin/sambaMonitor all"`

PidFiles	A list of PID files that contain the process ID (PID) of the processes that you want monitored and cleaned. These files 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—see the example below.
	The process ID can change when the process restarts. If the application takes time to update the PID file, the agent's monitor script may return an incorrect result. If this occurs, increase the ToleranceLimit in the resource definition.
	◆ Type and dimension: string-vector
	◆ Example:
	`{ "/var/lock/samba/smbd.pid" }`
	◆ Solaris 10—example in a global zone:
	`{ "/var/lock/samba/smbd.pid" }`
	◆ Solaris 10—example in a non-global zone:
	`{ "$zoneroot/var/lock/samba/smbd.pid" }`
	Where the *$zoneroot* 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 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"`

Sample Configurations

Sample 1

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

Sample 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. Also, the agent 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" }
)
```

Sample 3

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 specified by the `pid smbd.pid`, 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" }
    ContainerName = "zone1"
)
```


Error Messages

Message	Tag	Description
`File ${VCS_HOME}/bin/Application/functions not found.`	A	Check if this file exists and has proper permissions: `${VCS_HOME}/bin/Application/functions`
`Clean Program does not exist or is not executable.`	B	Check if the specified CleanProgram exists and the specified user has permission to execute it.
`Monitor Program does not exist or is not executable.`	B	Check if the specified MonitorProgram exists and the specified user has permission to execute it.
`None of the parameters (MonitorProgram, PidFiles, MonitorProcesses) are specified to monitor the application.`	B	Nothing is specified to monitor the resource. Specify at least one parameter to monitor the resource.
`No Start Program specified.`	B	No Start Program is specified in the resource definition and hence cannot bring the resource online.
`No Stop Program specified.`	B	No Stop Program is specified in the resource definition and hence cannot take the resource offline.
`Start Program does not exist or is not executable.`	B	Check if the specified StartProgram exists and the user specified permission to execute it.
`Stop Program does not exist or is not executable.`	B	Check if the specified StopProgram exists and the specified user has permission to execute it.
`User` *`user`* `does not exist.`	B	Check if the specified user exists on the local systems.

Process Agent

Starts, stops, and monitors a user-specified process.

Entry Points

- Online—Starts the process with optional arguments.
- Offline—Terminates the process with a `SIGTERM`. If the process does not exit, VCS sends a `SIGKILL`.
- Monitor—Checks to see if the process is alive by scanning the process table for the name of the executable pathname and argument list.

Type Definition

```
type Process (
        static str ArgList[] = { ContainerName, PathName, Arguments }
        str ContainerName
        str PathName
        str Arguments
)
```

Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
PathName	Defines 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. Pathname must not exceed 80 characters. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/lib/sendmail"`

Optional Attribute

Optional Attribute	Description, Type and Dimension, Default, and Example
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. Arguments 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"`

Sample Configurations

Sample 1

```
Process  usr_lib_sendmail (
  PathName = "/usr/lib/sendmail"
  Arguments = "bd q1h"
  )
```

Sample 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

Starts and monitors a process specified by the user.

Entry Points

- 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.

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
)
```

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
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"`
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`

Required Attribute	Description, Type and Dimension, Default, and Example
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. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/lib/nfs/nfsd"`

Optional Attribute

Optional Attribute	Description, Type and Dimension, Default, and Example
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: `"-a 8"`

Sample Configurations

Sample 1

```
ProcessOnOnly nfs_daemon(
  PathName = "/usr/lib/nfs/nfsd"
  Arguments = "-a 8"
  )
```


Sample 2

```
include "types.cf"

cluster ProcessCluster (
.
.
.
group ProcessOnOnlyGroup (
  SystemList = { sysa, sysb }
  AutoStartList = { sysa }
  )

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

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

// resource dependency tree
//
//      group ProcessOnOnlyGroup
//      {
//      ProcessOnOnly Process1
//      ProcessOnOnly Process2
//      }
```

Infrastructure and Support Agents 6

This chapter contains the following agents:

CampusCluster Agent

Uses Volume Manager (VM) mirroring as a data mobility solution in a clustered environment for disaster recovery. The CampusCluster agent causes a fast mirror re-synch (FMR) to remote plexes that have experienced a temporary downtime and then re-connected.

Caution To use VM for a campus cluster, you must have expert knowledge of Volume Manager and VCS.

For more information on using this agent, see the *VCS User's Manual*.

Requirements

A soft requirement, from the clustered host's standpoint, is that you might want to distinguish the physical location of each disk either by controller number or enclosure name.

Several hard requirements exist for using this agent:

- You must have a single VCS cluster with at least one node in each of two sites, where the sites are separated by a physical distance of no more than 80 kilometers.
- All volumes that have application-required data must be mirrored across site boundaries, with at least one plex in each site.
- You must disable Volume Manager's Relocation Daemon. Otherwise in case of a temporary site outage, all plexes locate to the same site.

Limitations

Global Cluster Option is *not* supported.

Entry Point

Monitor—Parses output from `vxnotify` to determine when lost disks have returned. Upon determining that a site has been restored, initiates the steps to re-synch the lost disks if possible.

State Definitions

Online—Resource of type ECC is always on.

Type Definition

```
type Campus (
     static str ArgList[] = { DiskGroup, RemoteCtlr }
     static str Operations = None
     str DiskGroup
     str RemoteCtlr
)
```

Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
DiskGroup	A string representing the name of the disk group to monitor. ◆ Type and dimension: string-scalar ◆ Example: `"disk_group"`

Optional Attribute

Optional Attribute	Description, Type and Dimension, Default, and Example
RemoteCtlr	Set this attribute if different controllers manage the disks at different sites from the standpoint of a host. You might need to localize this attribute if the hosts in the cluster have different controllers that manage the remote disks. The value should be of the format "`c#`" where `#` is the controller number, e.g. `c2`. Setting this attribute minimizes the number of disks that need to be rescanned when a path returns and is for performance only. If a host has a single controller or if enclosure based naming is used do not set this attribute. ◆ Type and dimension: string-scalar ◆ Example: `"c2"`

DNS Agent

The DNS agent updates the canonical name (CNAME) mapping in the domain name server when failing over applications across subnets (performing a wide area failover.)

If your failover target and source nodes are on the same subnet, then you do not need to use the DNS resource.

If, however, the failover target and source nodes reside on different subnets, you need to use the DNS agent. The agent updates the name server and allows clients to connect seamlessly to the failed over instance of the application service.

Entry Points

- Monitor—If the online lock file exists, monitor queries the name servers for the CNAME record for the alias and reports back ONLINE if the response from at least one of the name servers contains the same canonical name associated with the alias as specified in the HostName attribute. If not, the monitor reports the resource as OFFLINE.
- Online—Queries the authoritative name server of the domain for CNAME records and updates the CNAME record on the name server with the specified alias to canonical name mapping. It adds a new CNAME record if a related record is not found. Creates an online lock file if online was successful.
- Offline—Removes the online lock file, which the online entry point created.
- Open—Removes the online lock file if the online lock file exists, and the CNAME record on the name server does not contain the expected alias or canonical name mapping.
- Clean—Removes the online lock file, if present.

Type Definition

```
type DNS (
        static str ArgList[] = { Domain, Alias, Hostname, TTL,
        TSIGKeyFile, StealthMasters }
        str Domain
        str Alias
        str Hostname
        int TTL = 86400
        str TSIGKeyFile
        str StealthMasters[]
)
```

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
Domain	A string representing the domain name. ◆ Type and dimension: string-scalar ◆ Example: `"veritas.com"`
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.veritas.com`.
HostName	A string representing canonical name of a system or IP address. ◆ Type and dimension: string-scalar ◆ Example: `"mtv.veritas.com"`
TTL	A non-zero integer representing the “Time To Live” value, in seconds, for the DNS entries in the zone you are updating. A lower value means more hits on your DNS server, while a higher value means more time for your clients to learn about changes. ◆ Type and dimension: integer-scalar ◆ Default: `86400` ◆ Example: `3600`
StealthMasters	The list of primary master name servers in the domain. This is optional if the zone’s name server record lists the primary master name server. If the primary master name server is a stealth server, you must define this attribute. A stealth server is a name server that is authoritative for a zone, but is not listed in that zone’s NS records. ◆ Type and dimension: string-keylist ◆ Example: `{ "10.190.112.23" }`

Online Query

If the canonical name in the response CNAME record does not match the one specified for the resource, online tries to update the CNAME record on all authoritative master name servers in its domain (those master name servers that it can reach and where it has update permission). If the DNS update was successful, or was not necessary on at least one of the name servers, the online function creates an online lock file. The monitor entry point checks for the existence of this file. The online entry point does not create the online lock file if it is unable to update at least one domain name server.

A stealth server is a name server that is authoritative for a zone, but is not listed in that zone's NS records. If you specify the StealthMasters attribute, the online entry point tries to update the name servers specified in the StealthMasters attribute.

In BIND 8 and above, the primary master name server on receiving an update sends notification (NOTIFY) to all its slave servers asking them to pick up the update.

Monitor Scenarios

This table shows the various monitor scenarios:

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

Note The DNS agent supports BIND version 8 and above.

Sample Configuration

Take the VERITAS corporate web server as an example. A person using a web browser specifies the URL `www.veritas.com` to view the VERITAS web page, where `www.veritas.com` maps to the canonical name `mtv.veritas.com`, which is a host in Mountain View running the web server. The browser, in turn, retrieves the IP address for the web server by querying the domain name servers. If the web server for `www.veritas.com` is failed over from Mountain View to Heathrow, the domain name servers need to be updated with the new canonical name mapping so that the web browsers are directed to Heathrow instead of Mountain View. In this case, the DNS agent should update the name server to change the mapping of `www.veritas.com`, from `mtv.veritas.com` to the canonical name of the standby machine in Heathrow, `hro.veritas.com`, in case of a failover.

Sample Configuration

This is a DNS sample configuration.

```
DNS www (
Domain = "veritas.com"
Alias = www
Hostname = mtv
)
```

Bringing the `www` resource online updates the authoritative nameservers for domain `veritas.com` with the following CNAME record:

```
www.veritas.com. 86400 IN CNAME mtv.veritas.com
```

Thus all DNS lookups for `www.veritas.com` get resolved to `mtv.veritas.com`.

Secure DNS Update

The DNS agent by default—when the attribute 'TSIGKeyFile' is unspecified—expects the IP address of the hosts that can update the DNS records dynamically, to be specified in the `allow-updates` field of the zone. However, since IP addresses can be easily spoofed, a secure alternative is to use TSIG (Transaction Signature) as specified in RFC 2845. TSIG is a shared key message authentication mechanism available in DNS. A TSIG key provides a means to authenticate and verify the validity of DNS data exchanged, using a shared secret key between a resolver and either one or two servers.

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

▼ To use secure updates using TSIG keys

1. Run the `dnskeygen` command with the HMAC-MD5 (`-H`) option to generate a pair of files that contain the TSIG key:

   ```
   # dnskeygen -H 128 -h -n veritas.com.
     Kveritas.com.+157+00000.key
     Kveritas.com.+157+00000.private
   ```

2. Open either file. The contents of the file should look similar to:

   ```
   veritas.com. IN KEY 513 3 157 +Cdjlkef9ZTSeixERZ433Q==
   ```

3. Copy the shared secret (the TSIG key), which should look similar to:
   `+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 veritas.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-updates` substatement to reference the key:

   ```
   allow-updates { key veritas.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 = "veritas.com"
   Alias = www
   Hostname = north
   TSIGKeyFile = "/var/tsig/Kveritas.com.+157+00000.private"
   )
   ```

ElifNone Agent

Monitors a file.

Entry Point

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

Type Definition

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

Required Attribute

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

Sample Configuration

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


FileNone Agent

Monitors a file.

Entry Point

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

Type Definition

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

Required Attribute

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

Sample Configuration

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

FileOnOff Agent

Creates, removes, and monitors files.

Entry Points

- Online—Creates an empty file with the specified name if one 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.

Type Definition

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

Required Attribute

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

Sample Configuration

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


FileOnOnly Agent

Creates and monitors files.

Entry Points

- 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 agent reports as OFFLINE.

Type Definition

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

Required Attribute

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

Sample Configuration

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

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.

Note The attributes of the NotifierMngr agent cannot be dynamically changed using the `hares -modify` command. Changes made using this command are effective after notifier is restarted.

Entry Points

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

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.

Type Definition

```
type NotifierMngr (
        static int RestartLimit = 3
        static str ArgList[] = { EngineListeningPort, MessagesQueue,
        NotifierListeningPort, SnmpdTrapPort, SnmpCommunity,
        SnmpConsoles, SmtpServer, SmtpServerVrfyOff,
        SmtpServerTimeout, SmtpReturnPath,
        SmtpFromPath, SmtpRecipients }
        int EngineListeningPort = 14141
        int MessagesQueue = 30
        int NotifierListeningPort = 14144
        int SnmpdTrapPort = 162
        str SnmpCommunity = "public"
        str SnmpConsoles{}
        str SmtpServer
        boolean SmtpServerVrfyOff = 0
        int SmtpServerTimeout = 10
        str SmtpReturnPath
        str SmtpFromPath
        str SmtpRecipients{}
)
```

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
SnmpConsoles	Specifies the machine name of the SNMP manager and the severity level of the messages to be delivered to the SNMP manager. 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 SmtpServer is not specified; otherwise, SnmpConsoles is an optional attribute. The user can specify both SnmpConsoles and SmtpServer if necessary. ◆ Type and dimension: string-association ◆ Example: `{ "172.29.10.89" = Error, "172.29.10.56" = Information }`
SmtpServer	Specifies the machine name of the SMTP server. **Note** SmtpServer is a required attribute if SnmpConsoles is not specified; otherwise, SmtpServer is an optional attribute. The user can specify both SmtpServer and SnmpConsoles if necessary. ◆ Type and dimension: string-scalar ◆ Example: `"smtp.your_company.com"`

Optional Attributes

Optional Attribute	Description, Type and Dimension, Default, and Example
MessagesQueue	Size of the VCS engine's message queue. Its minimum value is 30. ◆ Type and dimension: integer-scalar ◆ Default: `30`
NotifierListeningPort	Any valid, unused TCP/IP port numbers. ◆ Type and dimension: integer-scalar ◆ Default: `14144`
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`
SnmpCommunity	Specifies the community ID for the SNMP manager. ◆ Type and dimension: string-scalar ◆ Default: `"public"`

SmtpRecipients	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 SmtpRecipients is a required attribute if you specify SmtpServer.
	◆ Type and dimension: string-association
	◆ Example:
	`{ "james@veritas.com" = SevereError,`
	`"admin@veritas.com" = Warning }`

SmtpServerVrfyOff	Setting this value to 1 results in the notifier not sending a SMTP VRFY request to the mail server specified in SmtpServer attribute while sending emails. Set this value to `1` if your mail server does not support SMTP VRFY command. ◆ Type and dimension: boolean-scalar ◆ Default: `0`

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`
SmtpReturnPath	Set to a valid email address, if you want the notifier to use a custom email address in the Return-Path: <> field. **Note** 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@veritas.com"`
SmtpFromPath	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@veritas.com"`
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`

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 "Phantom Agent" on page 96 for more information on verifying the status of groups that only contain OnOnly or Persistent resources (such as the NIC resource). NicGrp must be enabled 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 SnmpConsole (`snmpserv`). In this example, only messages of SevereError level are sent to the SmptServer (`smtp.your_company.com`), and the recipient (`vcsadmin@your_company.com`).

```
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 }
  SmtpServer = "smtp.your_company.com"
  SmtpRecipients = { "vcsadmin@your_company.com" = SevereError }
  )

ntfr requires nicproxy



// resource dependency tree
//
//      group Grp1
//      {
//      NotifierMngr ntfr
//              {
//              Proxy nicproxy
//              }
//      }
```


Phantom Agent

Enables VCS to determine the status of parallel service groups that do not include OnOff resources (resources that VCS can start and stop as required). Without the dummy resource provided by this agent, VCS cannot assess the status of groups that only contain None (Persistent) and OnOnly resources because the state of these resources is not considered in the process of determining whether a group is online. Refer to the *VERITAS Cluster Server User's Guide* for information on categories of service groups and resources.

Entry Point

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

Type Definition

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

Note The Dummy attribute is for VCS use only and is not configurable.

Sample Configurations

Sample 1

```
Phantom (
  )
```

Sample 2

The following example shows a complete configuration file (`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
  //      }
```


Proxy Agent

Mirrors the state of another resource on a local or remote system. Provides a means to specify and modify one resource and have it reflected by its proxies.

Entry Point

Monitor—Determines status based on the target resource status.

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
)
```

Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
TargetResName	Name of the target resource whose status is mirrored by Proxy resource. The target resource must be in a different resource group than the Proxy resource. ◆ Type and dimension: string-scalar ◆ Example: `"tmp_VRTSvcs_file1"`

Optional Attribute

Optional Attribute	Description, Type and Dimension, Default, and Example
TargetSysName	Mirror the status of the TargetResName on system specified by the TargetSysName variable. If this attribute is not specified, the Proxy resource assumes the system is local. ◆ Type and dimension: string-scalar ◆ Example: `"sysa"`

Sample Configurations

Sample 1

```
// Proxy resource to mirror the state of the resource
// tmp_VRTSvcs_file1 on the local system.

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

Sample 2

```
// Proxy resource to mirror the state of the resource
// tmp_VRTSvcs_file1 on sysa.

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


Sample 3

```
// Proxy agent to mirror the state of the resource mnic on
// the local system; note that target resource is in grp1,
// proxy in grp2; a target resource and its proxy cannot be in
// the same group.

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

  MultiNICA mnic (
    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
```

ServiceGroupHB Agent

Starts, stops, and monitors disk-based heartbeats associated with service groups. See the *VERITAS Cluster Server 4.1 User's Guide* for details.

The heartbeat region resides on a block device partition and consists of 128 blocks starting on the specified block number (see Disks attribute). The local system, via the ServiceGroupHB agent, tries to obtain "ownership" of the available disks as specified by the Disks' attribute. The system gains ownership of a disk when it determines that the disk is available and not owned by another system.

When the system's disk ownership meets the requirement of the AllOrNone attribute, it brings the resource online and monitors the resource. If the disk ownership falls below the AllOrNone requirement, VCS tries to fail over the group to another system.

Entry Points

- Online—Brings resource online after ownership of the required number of disks is obtained.
- Offline—Takes resource offline after relinquishing ownership of previously acquired disks.
- Clean—Takes resource offline and relinquishes ownership of previously acquired disks.
- Open—Creates logical disk objects based on Disks attribute at VCS startup.
- Close—At VCS shutdown, deletes the logical disk objects created by Open.
- Monitor—Periodically checks if local system has ownership of required number of disks.

Type Definition

```
type ServiceGroupHB (
        static str ArgList[] = { Disks, AllOrNone }
        static int OnlineRetryLimit = 5
        str Disks[]
        boolean AllOrNone = 1
)
```


Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
Disks	Specifies, in paired values, the disk partition and the starting block location to use for the heartbeat. Note that a block device partition is used for the disk heartbeating; for example, `/dev/dsk/c1t2d0s3`. A block device partition containing one or more heartbeat regions cannot be used for any other purpose. If the same partition is used for more than one heartbeat region, starting block numbers must be at least 64K (128 disk blocks) apart. ◆ Type and dimension: string-vector ◆ Example: `{ "c1t2d0s4" , "64" , "c2t2d0s4" , "64" }`
AllOrNone	Specifies number of disks for which "ownership" is required to bring the resource online, where: all available disks (`AllOrNone = 1`) and a simple majority of available disks (`AllOrNone = 0`). ◆ Type and dimension: boolean-scalar ◆ Default: `1`

Sample Configuration

In this example, the disk partitions `c1t2d0s4`, `c2t2d0s4`, and `c3t2d0s4` have service group heartbeat regions beginning at block 64 for service group `groupz`. The disk partition `c1t2d0s4` has a second heartbeat region beginning at block 192 for service group `groupy`.

The AllOrNone attribute is set to `0` for `sghb1`, specifying that the service group can come online with ownership of two disks.

```
.
system sysa
.
system sysb
.
.
.
group groupz (
.
.
        )

        ServiceGroupHB sghb1 (
                Disks = { c1t2d0s4, 64, c2t2d0s4, 64, c3t2d0s4, 64 }
```

```
                AllorNone = 0
                )

        Mount exp1
                MountPoint = "/soup"
                BlockDevice = "/dev/dsk/c1t2d0s5"
                FSType = ufs
                MountOpt = rw
                )

group groupy (
.
.
        )
.
        ServiceGroupHB sghb2 (
                Disks = { c1t2d0s4, 192 }
                )

        Mount exp2
                MountPoint = "/nuts"
                BlockDevice = "dev/dsk/c1t2d0s6"
                FSType = ufs
                MountOpt = rw
                )
.

exp1 requires sghb1
exp2 requires sghb2

// resource dependency tree
//
//
//      group groupz
//      {
//      Mount exp1
//              {
//              ServiceGroupHB sghb1
//              }
//      }
//      group groupy
//      {
//      Mount exp2
//              {
//              ServiceGroupHB sghb1
//              }
//      }
```


VRTSWebApp Agent

Brings Web applications online, takes them offline, and monitors their status. 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. This agent is used to monitor the Web Consoles of various VERITAS products, such as VCS and VVR.

Entry Points

- 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.

Type Definition

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

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
AppName	Name of the application as it appears in the Web server. Access the applications at: `http://localhost:8181/vcs`. ◆ Type and dimension: string-scalar ◆ Example: For VCS, use `vcs`.
InstallDir	Path to the Web application installation. The Web application must be installed as a `.war` file with the same name as the AppName parameter; the `vcs` application must be installed as `vcs.war`. This attribute should point to the directory that contains this `.war` file. ◆ Type and dimension: string-scalar ◆ Example: If AppName is `vcs` and InstallDir is `/opt/VRTSweb/VERITAS`, the agent constructs the path for the Web application as: `"/opt/VRTSweb/VERITAS/vcs.war"`
TimeForOnline	The time the Web application takes to start after it is loaded 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`

Sample Configuration

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


Zone Agent

Brings the agent online, takes it offline, monitors, and cleans Solaris 10 zones.

Entry Points

- 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—Another attempt to bring down a Solaris 10 zone forcefully.

Type Definition

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

Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
ZoneName	Name of the zone • Type and dimension: string-scalar • Example: `"localzone1"`

Optional Attribute

Optional Attribute	Description, Type and Dimension, Default, and Example
ShutDownGracePeriod	Allows the root user to set the number of seconds before shut down takes place. • Type and dimension: integer-scalar • Example: `10`

Sample Configuration

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


Index