VERITAS™

VERITAS Cluster Server 4.1

Bundled Agents Reference Guide

Solaris x64 Platform Edition

N18230F

November 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 47 presents Network File System (NFS) agent and the Share agent.

Chapter 5, "Services and Applications Agents" on page 53 presents the Application and Process agents. It describes the agents that make generic services and other applications highly available.

Chapter 6, "Infrastructure and Support Agents" on page 63 presents agents, such as the VRTSWebApp and ServiceGroupHB 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 xv.
#	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 as `bge0` does not require quotes as it contains only letters and numbers. Enclosing the string in double quotes is also acceptable—`"bge0"`. ◆ 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{}`.

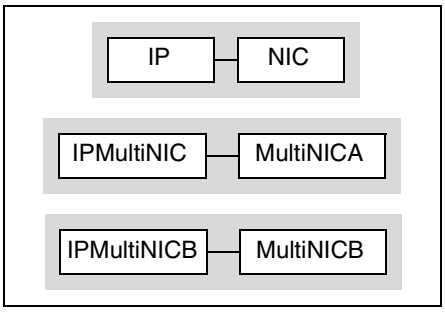
Network Agents 2

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: `"bge0"`

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 = bge0
  Address = "192.203.47.61"
  )
```

Sample 2: NetMask in decimal (base 10)

```
IP   IP_192_203_47_61 (
  Device = bge0
  Address = "192.203.47.61"
  NetMask = "255.255.248.0"
  )
```

Sample 3: NetMask in hexadecimal (base 16)

```
IP   IP_192_203_47_61 (
  Device = bge0
  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. Refer to the NIC's documentation for more information.

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: "bge0"

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_bge0 (
    Device = bge0
    PingOptimize = 1
    )
```

With Network Hosts

```
  NIC groupx_bge0 (
    Device = bge0
    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 = { bge0 = "10.128.8.42", e1000g0 = "10.128.8.42" }
    Device@sysb = { bge0 = "10.128.8.43", e1000g0 = "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: `bge0 = "10.128.8.42", e1000g0 = "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, `bge0` (`10.128.2.5`) and `e1000g0` (`10.128.2.8`), and you configure a third NIC, `e1000g1`, as the backup NIC to `bge0`. The agent does not fail over from `bge0` to `e1000g1` because all ping tests are redirected through `e1000g0` 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 `bge0` 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   bge0
224.0.0.0         10.128.8.42           U     1     0      bge0
default           10.128.8.42           UG    1     2402   bge0
127.0.0.1         127.0.0.1             UH    54    44249  lo0
```

If the RouteOptions attribute is not set and `bge0` fails, the MultiNICA agent migrates the base IP address to another NIC (such as `e1000g0`). The default route is no longer configured because it was associated with `bge0`. The display resembles:

```
Destination       Gateway             Flags Ref   Use    Interface
----------------- ------------------- ----- ----- ------ ---------
10.0.0.0          10.128.8.42           U     1    2408    e1000g0
224.0.0.0         10.128.8.42           U     1    0       e1000g0
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, `bge0` and `e1000g0`. The two interfaces, `bge0` and `e1000g0`, 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 `bge0` to `e1000g0`. If `e1000g0` fails, the address fails back to `bge0` if `bge0` 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 = { bge0 = "10.128.8.42", e1000g0 = "10.128.8.42" }
    Device@sysb = { bge0 = "10.128.8.43", e1000g0 = "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 = "e1000g0"`
	◆ 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 = "e1000g0" , "e1000g1"`
	◆ Example:
	`Device = "e1000g0" = 0, "e1000g1" = 2, "e1000g2" = 3`
	In this example, the MultiNICB agent uses interfaces `e1000g0`, `e1000g1`, and `e1000g2`. 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, `e1000g0`)
- 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 cluster resource 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 e1000g0 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 e1000g1:1 to interface e1000g0:1`

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.e1000g0`, add the following two lines:

```
north-e1000g0 netmask + broadcast + deprecated -failover up \
    addif north netmask + broadcast + up
```

Where **`north-e1000g0`** is the test IP address that the agent uses to determine the state of the `e1000g0` network card.

In the file `/etc/hostname.e1000g1`, add the following line:

```
north-e1000g1 netmask + broadcast + deprecated -failover up
```

Where **`north-e1000g1`** is the test IP address that the agent uses to determine the state of the `e1000g1` network card.

In the above example, `north-e1000g0` and `north-e1000g1` 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 = { e1000g0, e1000g1 }
  Device @south = { e1000g0, e1000g1 }
  NetworkHosts = { "192.1.0.1" }
)
Phantom gnic_p (
)
```

Storage Agents 3

This chapter contains the following agents:

- "DiskGroup Agent" on page 38
- "Mount Agent" on page 43
- "Volume Agent" on page 41

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
  )
```