VERITAS™

VERITAS Cluster Server 4.1

Bundled Agents Reference Guide

HP-UX

N12190G

June 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

Data Encryption Standard (DES)

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

SNMP Software

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

Contents

Preface

This guide provides reference information for the VCS agents bundled with VERITAS Cluster Server (VCS) software on the HP-UX 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 35, presents storage agents, such as the Mount and Volume agents.

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

Chapter 5, "Services and Applications Agents" on page 59, 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 71, presents agents, such as the DNS and NotifierMngr 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 stop VCS on all nodes of the cluster. First start VCS first on the node where you have made changes, and then the other nodes of the cluster.

Attributes

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

Attribute Data Types

Data Type	Description
string	Enclose strings, which are a sequence of characters, in double quotes ("). You do not have to enclose strings in quotes when they begin with a letter, and contains only letters, numbers, dashes (-), and underscores (_). ◆ A string defining a network interface such as `lan0` does not require quotes as it contains only letters and numbers. Enclosing the string in double quotes is also acceptable—`"lan0"`. ◆ 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 backward slashes (`\\`).
integer	Signed integer constants are a sequence of digits from `0` to `9`. You can precede them with a dash. They are to the 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. A comma separates each pair, for example: `{name=value, name1=value1}`. 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{}`.

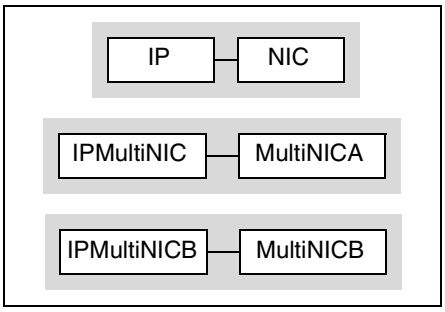
Network Agents 2

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 16 and the "IPMultiNIC Agent" on page 13 work together to make a virtual IP address, configured on servers with multiple adapters, highly available.
- The "MultiNICB Agent" on page 26 and the "IPMultiNICB Agent" on page 22 work together to make a virtual IP address, configured on servers with multiple adapters, highly available.

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

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.

VERITAS supports Auto-port Aggregation (APA) with the NIC and IP agents.

Entry Points

- Online—Plumbs the IP address to the NIC.
- 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.

State Definitions

- ONLINE—Indicates that the device is up and the specified IP address is assigned to the device.
- OFFLINE—Indicates that the device is down or the specified IP address is not assigned to the device.
- UNKNOWN—Indicates that the configuration is incorrect.

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: `"lan0"`

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 Example: `"255.255.210.0"`
Options	Options for the `ifconfig` command. ◆ Type and dimension: string-scalar ◆ Example: `"broadcast 192.203.15.255"`

Sample Configurations

Sample 1

```
IP IP_192_203_47_61 (
  Device = lan0
  Address = "192.203.47.61"
  )
```

Sample 2—NetMask in decimal (base 10)

```
IP   IP_192_203_47_61 (
  Device = lan0
  Address = "192.203.47.61"
  NetMask = "255.255.248.0"
  )
```

Sample 3—NetMask in hexadecimal (base 16)

```
IP   IP_192_203_47_61 (
  Device = lan0
  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.

VERITAS supports Auto-port Aggregation (APA) with the NIC and IP agents.

Before you use this agent, verify that the NIC has the correct administrative IP address and subnet mask.

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 configuration is incorrect.

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 ◆ "lan0"

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. ◆ Type and dimension: string-vector ◆ Example: NetworkHosts = { "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

With Network Hosts

```
NIC groupx_lan0 (
  Device = lan0
  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 any one interface that is configured in the MultiNICA resource.
- Offline—Removes the virtual IP address from the interface where the virtual IP address is configured.
- Monitor—Checks if the virtual IP address is configured on any one interface that is configured in the MultiNICA resource.
- Clean—Removes a virtual IP address from the interface where the virtual IP address is configured.
- Open—Initializes the setup that the agent uses to start in a clean state.
- Close—Cleans up the setup that the agent uses.

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 configuration is incorrect.

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: "192.205.10.14"
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
Options	Options for the ifconfig command. ◆ Type and dimension: string-scalar ◆ Example: "broadcast 192.203.15.255"

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 20.

```
group grp1 (
  SystemList = { sysa, sysb }
  AutoStartList = { sysa }
  )
  MultiNICA mnic (
    Device@sysa = { lan0 = "192.205.8.42", lan3 = "192.205.8.42" }
    Device@sysb = { lan0 = "192.205.8.43", lan3 = "192.205.8.43" }
    NetMask = "255.255.255.0"
    ArpDelay = 5
    Options = "broadcast 192.203.15.255"
    )

  IPMultiNIC ip1 (
    Address = "192.205.10.14"
    NetMask = "255.255.255.0"
    MultiNICResName = mnic
    Options = "broadcast 192.203.15.255"
    )

ip1 requires mnic

group grp2 (
  SystemList = { sysa, sysb }
  AutoStartList = { sysa }
  )

  IPMultiNIC ip2 (
    Address = "192.205.9.4"
    NetMask = "255.255.255.0"
    MultiNICResName = mnic
    Options = "broadcast 192.203.15.255"
    )
  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.

State Definitions

- ONLINE—Indicates that one or more of the network interfaces listed in the Device attribute of the resource is in working condition.
- FAULTED—Indicates that all of the network interfaces listed in the Device attribute failed.
- UNKNOWN—Indicates that the configuration is incorrect.

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: `{ lan0 = "192.205.8.42", lan3 = "192.205.8.42" }`

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. 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.
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`
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 can cause 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. ◆ Type and dimension: string-vector ◆ Example: `{ "166.93.2.1" , "166.97.1.2" }`
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
Options	The `ifconfig` options for the base IP address. ◆ Type and dimension: string-scalar ◆ Example: `"broadcast 192.203.15.255"`

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 192.98.16.103 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 one active NIC, `lan0(10.128.2.5)`, and you configure a second NIC, `lan1`, as the backup NIC to `lan0`. The agent does not fail over from `lan0` to `lan1` because all ping tests are redirected through `lan0` 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.

Using RouteOptions

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

For example, if the default gateway and `lan0` are both set to `11.236.99.248`, the output of the netstat `-rn` command from the routing table resembles:

```
Destination     Gateway         Flags  Refs  Interface  Pmtu
127.0.0.1       127.0.0.1       UH     0     lo0        4136
11.236.99.248   11.236.99.248   UH     0     lan0       4136
11.236.98.0     11.236.99.248   U      2     lan0       1500
127.0.0.0       127.0.0.1       U      0     lo0           0
default         11.236.99.248   UG     0     lan0          0
```

If the RouteOptions attribute is not set and `lan0` fails, the MultiNICA agent migrates the base IP address to another NIC (such as `lan1`). The default route is no longer configured because it was associated with `lan0`. The display from the routing table resembles:

```
Destination     Gateway         Flags  Refs  Interface  Pmtu
127.0.0.1       127.0.0.1       UH     0     lo0        4136
11.236.99.161   11.236.99.161   UH     0     lan2       4136
11.236.98.0     11.236.99.161   U      2     lan2       1500
```

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

```
RouteOptions@sysa = "default 11.236.99.248 0"
RouteOptions@sysb = "default 11.236.99.249 0"
```

Sample Configuration: MultiNICA and IPMultiNIC

In the following example, two machines, sysa and sysb, each have a pair of network interfaces, `lan0` and `lan3`. The two interfaces, `lan0` and `lan3`, 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 `lan0` to `lan3`. If `lan3` fails, the address fails back to `lan0` if `lan0` 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 13.

```
group grp1 (
  SystemList = { sysa, sysb }
  AutoStartList = { sysa }
  )
  MultiNICA mnic (
    Device@sysa = { lan0 = "192.205.8.42", lan3 = "192.205.8.42" }
    Device@sysb = { lan0 = "192.205.8.43", lan3 = "192.205.8.43" }
    NetMask = "255.255.255.0"
    ArpDelay = 5
    Options = "broadcast 192.203.15.255"
    )

  IPMultiNIC ip1 (
    Address = "192.205.10.14"
    NetMask = "255.255.255.0"
    MultiNICResName = mnic
    Options = "broadcast 192.203.15.255"
    )

ip1 requires mnic


group grp2 (
  SystemList = { sysa, sysb }
  AutoStartList = { sysa }
  )

  IPMultiNIC ip2 (
    Address = "192.205.9.4"
    NetMask = "255.255.255.0"
    MultiNICResName = mnic
    Options = "broadcast 192.203.15.255"
    )
  Proxy proxy (
    TargetResName = mnic
    )

ip2 requires proxy
```


IPMultiNICB Agent

Works with the MultiNICB agent. Manages a virtual IP address configured as an alias on one of the interfaces of a MultiNICB resource. If the NIC where the logical IP address is configured is marked DOWN by the MultiNICB agent, or a FAILED flag is set on the interface, the resource is reported FAULTED. If multiple service groups have IPMultiNICB resources associated with the same MultiNICB resource, only one group should have the MultiNICB resource. The other groups should have a proxy resource pointing to it.

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 configuration is incorrect.

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: `"192.205.10.15"`
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`"

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 = "lan0"`
	◆ 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

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 32 for a sample configuration of IPMultiNICB and MultiNICB.

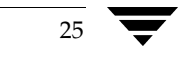
Manually Migrating a Logical IP Address

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

```
# haipswitch -s MultiNICB resname
# haipswitch MultiNICB_res_name IPMultiNICB_res_name ip_address
netmask from_physical_ip to_physical_ip
```

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.

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.

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.

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.
- FAULTED—Indicates that all of the network interfaces listed in the Device attribute failed.
- UNKNOWN—Indicates that the configuration is incorrect.

Type Definition

```
type MultiNICB (
        static int MonitorInterval = 10
        static int OfflineMonitorInterval = 60
        static int MonitorTimeout = 60
        static int Operations = None
        static str ArgList[] = { Device, NetworkHosts,
        LinkTestRatio, IgnoreLinkStatus, NetworkTimeout,
        OnlineTestRepeatCount, OfflineTestRepeatCount, NoBroadcast,
         DefaultRouter, Failback}
        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 = { "lan0" , "lan4" }`
	◆ Example:
	`Device = { "lan0" = 0, "lan1" = 2, "lan2" = 3 }`
	In this example, the MultiNICB agent uses interfaces `lan0`, `lan1`, and `lan2`. The MultiNICB agent passes on the associated interface aliases `0`, `2`, and `3` to the IPMultiNICB agent.

Optional Attributes

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. ◆ 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`

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. ◆ 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`

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 give 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.

✔ If NetworkHosts is specified, the hosts must be directly accessible on the LAN.

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, `lan0`)
- device's previous status (`0` for down, `1` for up)
- device's current status
- 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: Device lan0 status changed from DOWN to 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 lan1:1 to interface lan0:1`

Sample Configuration

VCS Configuration

The following is an example VCS configuration.

```
include "types.cf"

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 ipmnicb (
                BaseResName = mnicb
                Address = "192.1.0.201"
                NetMask = "255.255.0.0"
                DeviceChoice = 1
                )

        MultiNICB mnicb (
                Device @north = { lan0 = 0, lan4 = 1 }
                Device @south = { lan0 = 0, lan4 = 1 }
                NetworkHosts = { "192.1.0.1" }
                DefaultRouter = "0.0.0.0"
                )

ipmnicb requires mnicb
```


Storage Agents 3

This chapter contains the following agents:

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.
- Offline—Deports the disk group.
- Monitor—Determines if the disk group is online or offline. 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.
- 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 NumThreads = 1
  static int OnlineRetryLimit = 1
  static str ArgList[] = { DiskGroup, StartVolumes, StopVolumes,
  MonitorOnly, MonitorReservation, tempUseFence }
  str DiskGroup
  str StartVolumes = 1
  str StopVolumes = 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 4.1.

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 Configurations

Sample 1

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

Sample 2—DiskGroup, Volume, and Mount Dependencies

This sample configuration shows the DiskGroup, Volume, and Mount dependencies:

```
group sample_vxvm_group (
  SystemList = { System1, System2 }
  AutoStartList = { System1 }
  )

        Volume vres (
                Volume = vol1
                DiskGroup = dg2
                )

        Mount mres (
                MountPoint = "/dir1"
                BlockDevice = "/dev/vx/dsk/dg2/vol1"
                FSType = vxfs
                FsckOpt = "-y"
                )

        DiskGroup dres (
                DiskGroup = dg2
                StartVolumes = 0
                StopVolumes = 0
                )

  mres requires vres
  vres requires dres
```


Volume Agent

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

Entry Points

- Online—Starts the volume.
- Offline—Stops the volume.
- Monitor—Determines if the volume is online or offline by reading a block from the raw device interface to the volume.
- Clean—Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.

State Definitions

- ONLINE—Indicates that the specified volume is started and that I/O is permitted.
- OFFLINE—Indicates that the specified volume is not started—and I/O is not permitted.
- UNKNOWN—Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

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
  )
```

See "Sample 2—DiskGroup, Volume, and Mount Dependencies" on page 39 for more configurations.

LVMCombo Agent

Defines the logical volumes and volume groups associated with an application. Use LVMCombo as an alternative to LVMLogicalVolume and LVMVolumeGroup when defining logical volumes and volume groups. While LVMCombo is similar to LVMLogicalVolume and LVMVolumeGroup, it does not enable LVM configuration information to be backed up every time resources are brought online or taken offline.

Entry Points

- Online—Activates the volume group and any of the logical volumes that are not available. While each system in the cluster must import the volume group, each system does not need to activate it.

  This agent does not import volume groups because of the way LVM stores configuration information. Use the HP-UX SAM tool to import a volume group.
- Offline—Deactivates the volume group, but does not deactivate the logical volumes. The logical volumes are automatically deactivated when the volume group is deactivated.
- Monitor—If the volume group and all of the logical volumes are available, the resource is online. Otherwise, the resource faults.

Note The monitor entry point does not perform any I/O on disk. If a disk that makes up a logical volume is powered off, the agent is not aware of this situation until LVM marks the logical volume unavailable. This may occur if the file system or the application using the logical volume attempts an I/O operation and fails. LVM can then set the logical volume as unavailable.

State Definitions

ONLINE—Indicates that the Volume Group and Logical Volumes are active.

OFFLINE—Indicates that the Volume Group and Logical Volumes are not active.

UNKNOWN—Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

Type Definition

```
type LVMCombo (
        static str ArgList[] = { VolumeGroup, LogicalVolumes }
        str VolumeGroup
        str LogicalVolumes[]
)
```

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
LogicalVolumes	List of logical volumes in a volume group. ◆ Type and dimension: string-vector ◆ Example: `{ "lvol1" , "lvol2" }`
VolumeGroup	Name of a volume group. ◆ Type and dimension: string-scalar ◆ Example: `"vg01"`

Sample Configurations

Sample 1

```
LVMCombo vg01 (
  VolumeGroup = vg01
  LogicalVolumes = { lvol1, lvol2 }
  )
```


Sample 2—LVMCombo and Mount Dependencies

This sample configuration shows the LVMCombo and Mount dependencies:

```
group sample_lvmcombo (
  SystemList = { System1, System2 }
  AutoStartList = { System1 }
  )

  LVMCombo lvmcmbres (
          VolumeGroup = vg02
          LogicalVolumes = { lvol1 }
          )

  Mount mres (
          MountPoint = "/dir2"
          BlockDevice = "/dev/vg02/lvol1"
          FSType = vxfs
          MountOpt = ro
          FsckOpt = "-y"
          )

  mres requires lvmcmbres
```

LVMLogicalVolume Agent

Brings online, takes offline, and monitors Logical Volume Manager (LVM) logical volumes.

Entry Point

- Online—Activates the logical volume.
- Offline—Deactivates the logical volume.
- Monitor—Determines if the logical volume is accessible by performing read I/O on the raw logical volume.

State Definitions

ONLINE—Indicates that the Logical Volume is active.

OFFLINE—Indicates that the Logical Volume is not active.

UNKNOWN—Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

Type Definition

```
type LVMLogicalVolume (
        static str ArgList[] = { LogicalVolume, VolumeGroup }
        str LogicalVolume
        str VolumeGroup
)
```

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
LogicalVolume	Name of the logical volume. ◆ Type and dimension: string-scalar ◆ Example: `"lvol1"`
VolumeGroup	Name of a volume group containing the logical volume. ◆ Type and dimension: string-scalar ◆ Example: `"vg1"`

Sample Configuration

```
LVMLogicalVolume sharedg_lvol1 (
   LogicalVolume = lvol1
   VolumeGroup = sharevg
   )
```

See "Sample 2—LVMVolumeGroup, LVMLogicalVolume, and Mount Dependencies" on page 48 for more configurations.

LVMVolumeGroup Agent

Activates, deactivates, and monitors LVM volume groups.

Entry Points

- Online—Activates a volume group.While each system in the cluster must import the volume group, each system does not need to activate it.

  This agent does not import volume groups because of the way LVM stores configuration information. Use the HP-UX SAM tool to import a volume group.
- Offline—Deactivates a volume group with the `vgchange` command.
- Monitor—Determines whether the volume group is available.

State Definitions

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

Type Definition

```
type LVMVolumeGroup (
        static str ArgList[] = { VolumeGroup }
        str VolumeGroup
)
```

Required Attribute

Required Attribute	Description, Type and Dimension, Default, and Example
VolumeGroup	Name of the volume group configured with LVM. ◆ Type and dimension: string-scalar ◆ Example: `"sharevg"`

Sample Configurations

Sample 1

```
LVMVolumeGroup sharevg (
  VolumeGroup = sharevg
  )
```

Sample 2—LVMVolumeGroup, LVMLogicalVolume, and Mount Dependencies

This sample configuration shows the LVMVolumeGroup, LVMLogicalVolume, and Mount dependencies:

```
group sample_lvm (
  SystemList = { System1, System2 }
  AutoStartList = { System1 }
  )

        LVMLogicalVolume lvolres (
                LogicalVolume = lvol2
                VolumeGroup = vg01
                )

        LVMVolumeGroup lvgres (
                VolumeGroup = vg01
                )

        Mount mres (
                MountPoint = "/dir2"
                BlockDevice = "/dev/vg01/lvol2"
                FSType = vxfs
                MountOpt = ro
                FsckOpt = "-y"
                )

  mres requires lvolres
  lvolres requires lvgres
```

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.
- 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 either with the configuration or the ability to determine the status of the resource.

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"` ◆ Example: `"/dev/vg02/lvol1"`
FsckOpt	Options for `fsck` command. You must include `-y` or `-n` must as arguments to `fsck` for the resource to come online. The `-y` argument enables the VxFS file systems to perform a log replay before a full `fsck` operation. 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, hfs, 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:

```
bdf <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"
  )
```

For more configurations, see:

- "Sample 2—LVMCombo and Mount Dependencies" on page 44
- "Sample 2—LVMVolumeGroup, LVMLogicalVolume, and Mount Dependencies" on page 48
- "Sample 2—DiskGroup, Volume, and Mount Dependencies" on page 39

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.

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.
- 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, Protocol }
        static str Operations = OnOnly
        int Nservers = 4
        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: 4 ◆ Example: 24

Sample Configuration

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


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/xtab` file and looks for an entry for the file system specified by PathName. If the entry exists, monitor returns ONLINE.

State Definitions

- ONLINE—Indicates that specified directory is exported to the client.
- OFFLINE—Indicates that the specified directory is not exported to the client.
- UNKNOWN—Indicates that the agent could not determine the state of the resource or that the resource attributes are invalid.

Type Definition

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

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 ArgList[] = { User, StartProgram , StopProgram ,
    CleanProgram , MonitorProgram , PidFiles , MonitorProcesses }
        str User = "root"
        str StartProgram
        str StopProgram
        str CleanProgram
        str MonitorProgram
        str PidFiles[]
        str MonitorProcesses[]
)
```


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. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/sbin/sample_app 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. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/sbin/sample_app stop"`
At least one of the following attributes: ◆ MonitorProcesses ◆ MonitorProgram ◆ PidFiles	See "Optional Attributes" on page 62.

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. ◆ Type and dimension: string-scalar ◆ Example: `"/usr/sbin/sample_app force stop"`
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: `{ "sample_app_process" }`

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. 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/sbin/sample_app_monitor 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. 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: `"/etc/sample/sample_app.pid"`
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 sample_app (
  User = "root"
  StartProgram = "/usr/sbin/sample_app start"
  StopProgram = "/usr/sbin/sample_app stop"
  PidFiles = { "/etc/sample_app.pid" }
  MonitorProcesses = { "sample_app_process" }
)
```

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 sample_app2 (
  StartProgram = "/usr/sbin/sample_app start"
  StopProgram = "/usr/sbin/sample_app stop"
  CleanProgram = "/usr/sbin/sample_app force stop"
  MonitorProgram = "/usr/local/bin/sampleMonitor all"
  MonitorProcesses = { "sample_app_process" }
)
```

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, the agent 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[] = { PathName, Arguments, UserName, Priority,
  PidFile }
  str PathName
  str Arguments
  str UserName = root
  str Priority = 20
  str PidFile
)
```

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/sbin/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 -q30m"`
UserName	The user whose ID is used to run the process. The process along with the arguments must run the context of the specified user. Type and Dimension: string-scalar Default: `"root"` Example: `"user1"`
Priority	Priority with which the process runs. It is effective only when the user is root. Range is 0 to 39 where a process with a priority 0 is the highest. Type and Dimension: string-scalar Default: `"20"` Example: `"35"`
PidFile	File that stores process PID. ◆ Type and dimension: string-scalar ◆ Example: `"/etc/mail/sendmail.pid"`

Sample Configurations

Sample 1

```
Process sendmail1 (
  PathName = "/usr/sbin/sendmail"
  Arguments = "-bd -q30m"
  User = root
  Priority = 10
  PidFile = "/etc/mail/sendmail.pid"
)
```

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.

State Definition

- ONLINE—Process is running.
- FAULTED—Process is not running.
- UNKNOWN—Invalid configuration or agent unable to determine the state of the process.

Type Definition

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

Required Attributes

Required Attribute	Description, Type and Dimension, Default, and Example
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`
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/sbin/sendmail"`

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: `"-bd -q30m"`

Sample Configuration

```
ProcessOnOnly sendmail_pr (
PathName = "/usr/sbin/sendmail"
Arguments = "-bd -q30m"
)
```


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—CampusCluster resource type is always on.

Type Definition

```
type CampusCluster (
   static int NumThreads = 1
   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.

State Definitions

- ONLINE—Online lock exists and the CNAME RR is as expected.
- OFFLINE—Either the lock does not exist, or the expected record is not found.
- UNKNOWN—Problem with configuration.

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`

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

Optional Attribute

Optional Attribute	Description, Type and Dimension, Default, and Example
`TSIGKeyFile`	Specifies the absolute path to the file containing the private TSIG (Transaction Signature) key. For details, refer to "Secure DNS Update" on page 78. ◆ Type and dimension: string-scalar ◆ Example: `"/var/tsig/Kveritas.com.+157+00000.private"`

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` resolve 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

Checks for a file's absence.

Entry Point

- Monitor—Checks for the specified file. If it exists, the agent reports as FAULTED. 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

Checks for a file's existence.

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 FAULTED.

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 FAULTED.

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@example.com" = SevereError,`
	`"admin@example.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@example.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@example.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 91 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 (
    Device  = lan0
    NetworkHosts = { "166.93.2.1", "166.97.1.2" }
    )

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 = { lan0 = "192.98.16.103",lan3 = "192.98.16.103" }
    Device@sysb = { lan0 = "192.98.16.104",lan3 = "192.98.16.104" }
    NetMask = "255.255.255.0"
    ArpDelay = 5
    Options = "broadcast 192.203.15.255"
    RouteOptions@sysa = "default 192.98.16.103 0"
    RouteOptions@sysb = "default 192.98.16.104 0"
    )
  IPMultiNIC ip1 (
    Address = "192.98.14.78"
    NetMask = "255.255.255.0"
    MultiNICResName = mnic
    Options = "broadcast 192.203.15.255"
    )

ip1 requires mnic


group grp2 (
  SystemList = { sysa, sysb }
  AutoStartList = { sysa }
  )
  IPMultiNIC ip2 (
    Address = "192.98.14.79"
    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.

State Definitions

- ONLINE—Indicates the system has the ownership of the logical disk objects.
- OFFLINE—Indicates that the system does not own the logical disk objects.
- UNKNOWN—Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

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 block device (typically a logical volume) and the starting block location to use for the heartbeat. For example, if the block device /dev/vg01/lvol_hb is used for the heartbeat region, and the starting block is 16, the paired set of values is /dev/vg01/lvol_hb, 16. 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: `{ "/dev/vg01/hb_vol1" , "64" , "/dev/vg01/hb_vol2" }`
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 volumes `/dev/vg01/hb_vol1`, `/dev/vg01/hb_vol2`, and `/dev/vg01/hb_vol3` have service group heartbeat regions beginning at block 64 for service group `groupz`. The device `/dev/vg01/hb_vol1` 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 = { /dev/vg01/hb_vol1, 64, /dev/vg01/hb_vol2, 64,
                         /dev/vg01/hb_vol3, 64 }
                AllorNone = 0
                )

        Mount exp1
                MountPoint = "/soup"
                BlockDevice = "/dev/vg01/vol4"
                FSType = ufs
                MountOpt = rw
                )

group groupy (
.
.
        )
.
        ServiceGroupHB sghb2 (
                Disks = { /dev/ vg01/hb_vol1, 192 }
                )

        Mount exp2
                MountPoint = "/nuts"
                BlockDevice = "/dev/vg01/lvol5"
                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.

State Definitions

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

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
  )
```


Index

W