VERITAS Cluster Server™ 1.3.0 Bundled Agents

Reference Guide

Solaris

October 2000
30-000037-399

VERITAS

Disclaimer

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

Copyright

Printed in the USA, October 2000.

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

Contents

Preface

This document provides reference for the VCS agents bundled with the VCS software.

For information on purchasing VCS agents or other VERITAS products, contact your VERITAS sales representative:

U.S. and Canadian Customers: 1-800-327-2232

International Customers: +1 (407) 531-7501 or consult the Web (www.veritas.com) for the VERITAS sales office in your area.

Email: sales.mail@veritas.com

VCS Enterprise and Storage Agents

VCS enterprise and storage agents are not included with the VCS software, but are sold separately. Contact your VERITAS sales representative for details on these agents or additional agents under development:

- Informix
- NetApp (storage agent)
- NetBackup
- Oracle
- PCNetlink
- SuiteSpot
- Sun Internet Mail Server (SIMS)
- Sybase

Technical Support

For assistance with this VERITAS product, or for information regarding VERITAS service packages, contact Technical Support at 800.342.0652 (U.S. and Canada). You may also contact Technical Support via email at support@veritas.com.

For Customers Outside U.S. and Canada

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

Conventions

Typeface	Usage
`courier`	Computer output, files, attribute names, device names, and directories
`courier` (bold)	User input and commands, keywords in grammar syntax
italic	New terms, titles, emphasis
italic	Variables
Symbol	**Usage**
%	C shell prompt
$	Bourne/Korn shell prompt
#	Superuser prompt (for all shells)

Introduction 1

VCS Bundled Agents

VCS bundled agents are part of VCS, installed when VCS is installed. The bundled agents are VCS processes that manage predefined resource types. A system 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 is started, it obtains the necessary configuration information from VCS. It then periodically monitors the resources, and updates VCS with the resource status

Typically, VCS agents:

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

Resources and Their Attributes

Resources are the key parts of a system and are known by their type, such as a disk, a volume, an IP address, or a mount point. Resource types are defined in the `types.cf` file by a collection of attributes. The VCS configuration file, `main.cf`, which lists an include directive to the `types.cf` file, contains the values for the attributes of the resources.

A given value for a given attribute configures the resource to function in a specific way. By modifying the value of an attribute of a resource, you change the way the VCS agent manages the resource.

For example, the IP agent monitors an IP address resource. The specific address to be monitored, which is resource type “IP,” is identified by the attribute “Address” whose value is the specific IP address.

In Chapter two of this document, each VCS bundled agent is described. Tables include the definitions of the resource type and its modifiable attributes. The tables also indicate which attributes are required and which are optional. The resource type as it is defined in the `types.cf` file is included along with a sample of the configuration in `main.cf`.

Modifying the Agents and Their Resources

You can use the VCS GUI or the VCS command line to dynamically modify the configuration of the resources managed by an agent. See the *VERITAS Cluster Server User's Guide* for instructions on how to perform these tasks.

It is also possible to edit the `main.cf` file directly. To implement these changes, however, you must reboot your system to put the changes into effect.

A List of Bundled Agents

Agents included with VCS are referred to as *bundled* agents:

- Application
- Disk
- DiskGroup
- DiskReservation
- ElifNone
- FileNone
- FileOnOff
- FileOnOnly
- IP
- IPMultiNIC
- Mount
- MultiNICA
- NFS
- NIC
- Phantom
- Process
- Proxy
- ServiceGroupHB
- Share
- Volume

VCS Bundled Agents 2

Application Agent

Description	Brings applications online, takes them offline, and monitors their status. You may specify different executables for the online, offline, and the monitor routines. An application runs in the context of root by default. You may choose to run an application in a user context by specifying the user name. The agent starts and stops the application with user-specified programs. You can monitor the application in the following ways: ◆ Use the monitor program ◆ Specify a list of processes to be monitored ◆ Specify a list of process ID files to be monitored ◆ 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 no MonitorProgram is specified, the monitor routine verifies that all processes specified in PidFiles and MonitorProcesses are running. If the MonitorProgram is defined, the agent executes the MonitorProgram. ◆ Clean—Kills all processes specified in PidFiles or in MonitorProcesses. 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 that the state of the application being monitored is indeterminable.

Attributes	Type and Dimension	Definition
CleanProgram	string-scalar	The executable that forcibly stops the application. Note that you must specify the complete path of the executable. Command-line arguments, if used, follow the name of the executable, separated by spaces.
MonitorProcesses	string-vector	A list of processes to be monitored. Each process name is the name of an executable. You must qualify the executable name with its complete path if the path is to be used to start the executable. Note that the process name must be the name displayed by the **`ps -ef`** command for the process.
MonitorProgram	string-scalar	The executable that will monitor the application. Note that you must specify the complete path of the executable. Command-line arguments, if used, follow the name of the executable, separated by spaces. Note that the return value of MonitorProgram must be a VCSAgResState value: 0 = OFFLINE, 1 = ONLINE.
PidFiles	string-vector	A list of pid files that contain the process ID of the processes to be monitored. Note that these are application-generated files. You must specify the complete path of each pid file in the list. Note that the process ID of a process could 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.
StartProgram	string-scalar	The name of the executable that starts the application. Note that you must specify the complete path of the executable. Command-line arguments, if used, follow the name of the executable, separated by spaces.
StopProgram	string-scalar	The executable that stops the application. Note that you must specify the complete path of the executable. Command-line arguments, if used, follow the name of the executable, separated by spaces.
User	string-scalar	The user whose id is used to run the StartProgram, the StopProgram, the MonitorProgram, and the CleanProgram. Default is root.

Type Definition

```
type Application (
   static str ArgList[] = { User, StartProgram, StopProgram,
      CleanProgram, MonitorProgram, PidFiles,
      MonitorProcesses }
   str User
   str StartProgram
   str StopProgram
   str CleanProgram
   str MonitorProgram
   str PidFiles[]
   str MonitorProcesses[]
)
```

Sample Configurations

In the following example, the executable *samba* is configured as StartProgram and StopProgram, with *start* and *stop* specified as command-line arguments respectively. The agent is configured to monitor two processes: a process specified by the pid *smbd.pid*, and the process *nmbd*.

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

In the following example, no user is specified, so the root user will be used. The executable *samba* is used to start and stop the application, with *start* and *stop* as the command-line arguments respectively. The executable *sambaMonitor* monitors the application and uses *all* as its command-line argument. In addition, the processes *smbd* and *nmbd* are monitored.

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


Application Agent Error Messages

Message	Tag	Description
Error opening directory `/proc`.	B	The opendir call failed. **Solution:** Verify the file permissions.
Program *programname* does not execute or is not an executable.	B	The value for the attribute StartProgram, StopProgram, MonitorProgram, or CleanProgram is not a valid executable file. **Solution:** Verify that the attributes StartProgram, StopProgram, MonitorProgram, and CleanProgram are correctly defined in the configuration file.
Pid file does not exist.	B	One or more pid file paths specified in the PidFiles attribute is invalid.
Cannot open pid file.	B	One or more pid file paths specified in the PidFiles attribute could not be opened.
One or more specified processes are not running.	D	At least one process specified in PidFiles or in MonitorProcesses is not running.

Disk Agent

Description	Manages a raw disk.	
Entry Points	◆ Online—Not applicable. ◆ Offline—Not applicable. ◆ Monitor—Determines if disk is accessible by performing read I/O on raw disk.	
Required Attribute	**Type and Dimension**	**Definition**
Partition	string-scalar	Indicates which partition to monitor. Partition is specified with the full path beginning with a slash (/). Otherwise the name given is assumed to reside in /dev/rdsk.

Type Definition

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

Sample Configuration

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


DiskGroup Agent

Description	Brings online, takes offline, and monitors a VERITAS Volume Manager disk group.	
Entry Points	◆ Online—Using the command `vxdg`, this script imports the disk group. ◆ Offline—Using the command `vxdg`, this script deports the disk group. ◆ Monitor—Using the command `vxdg`, this agent determines if the disk group is online or offline. If disk group has been imported with `noautoimport=off`, and if the group is not frozen, the group to which the disk group belongs is taken offline.	
Required Attribute	**Type and Dimension**	**Definition**
DiskGroup	string-scalar	Disk group name.
Optional Attributes	**Type and Dimension**	**Definition**
StartVolumes	string-scalar	If value is 1, the DiskGroup `online` script starts all volumes belonging to that disk group after importing. Default is 1.
StopVolumes	string-scalar	If value is 1, the DiskGroup `offline` script stops all volumes belonging to that disk group before deporting. Default is 1.

Type Definition

```
type DiskGroup (
   static int OnlineRetryLimit = 1
   str DiskGroup
   NameRule = resource.DiskGroup
   static str ArgList[] = { DiskGroup, StartVolumes,
                            StopVolumes, MonitorOnly }
   str StartVolumes = 1
   str StopVolumes = 1
   static int NumThreads = 1
)
```

Sample Configuration

```
DiskGroup    sharedg (
   DiskGroup = sharedg
   )
```


DiskReservation Agent

Using the DiskReservation agent, a user can configure a node to reserve shared disks for an application. The reservations prevent disk data corruption by restricting other nodes from accessing and writing to the disks.

While the DiskReservation agent is implemented based on the SCSI II reservation feature, the automatic probing feature reestablishes reservations that are lost in the event the bus or disk is reset. The DiskReservation agent supports all SCSI II disks.

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

.

Description	Reserves and monitors disks for a system, enabling the resource to go online on that system when a group of disks is reserved. User specifies a list of raw disk devices. User may specify that all, or a percentage, of the accessible disks be reserved. An automatic probing feature allows systems to keep reservations even when disks or the bus are reset. The optional FailFast feature minimizes data corruption in the event of a reservation conflict by causing the system to panic.	
Entry Points	◆ Online—Onlines resource after reserving all or a specified percentage of accessible disks. ◆ Offline—Releases reservations on reserved disks. ◆ Monitor—Monitor the accessibility and reservation status of the reserved disks.	
Required Attribute	**Type and Dimension**	**Definition**
Disks	string-vector	A list of raw disk devices; the absolute or relative device path can be used. Note that the order of the disks in the list must be the same across all systems in the cluster, even if the same device has a different name on different systems.
Optional Attributes	**Type and Dimension**	**Definition**
FailFast	boolean-scalar	Optional. If enabled, FailFast causes the system to panic when a reservation conflict is detected, thereby reducing the chance of further data corruption. Default = 0.
ConfigPercentage	string-scalar	Optional. The minimum percentage of configured disks that can be reserved before resource can go online. Default is 80.
ProbeInterval	string-scalar	Optional. Alters the periodicity of the automatic probe function. A lower value for ProbeInterval specifies more frequent probes and provides for quicker discovery of reservation conflicts. The unit of measure is seconds. Default = 6.

When Not to Use DiskReservation Agent

The DiskReservation agent prevents data corruption by restricting all nodes except the node with reservations from accessing a shared disk. Therefore, it is not recommended to use the DiskReservation agent where it is desirable to use a shared device on multiple nodes, even if only one node is intended to have write access.

Type Definition

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

Sample Configurations

CASE 1: Configuring the mount resource to depend on the DiskReservation resource ensures that when a given disk is mounted on a node, no other node can mount the same disk. The group definition in `main.cf` would look like the following.

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

      DiskReservation dr (
         Disks = { c1t2d0s4 }
         FailFast = 1
         )

      Mount export1 (
         MountPoint = "/test"
         BlockDevice = "/dev/dsk/c1t2d0s4"
         FSType = ufs
         MountOpt = rw
         )

         export1 requires dr
```

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

CASE 2: The DiskReservation resource can also be used in conjunction with the Volume manager. In the following example of a group defined in the `main.cf` file, the Mount resource depends on the Volume resource, which depends on the DiskGroup resource, which depends on the DiskReservation resource.

Note that the DiskReservation resource's Disks list must contain the raw devices that constitute the disk group. You can use the VERITAS Volume Manager command "`vxdisk list`" to list the devices constituting the volume.

There are five disks in the user-defined group, four of which must be accessible and reserved for the resource group to go online (the default ConfigPercentage is 80).

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

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

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

   DiskGroup resdg (
      DiskGroup = resdg
      )

   Volume resvol (
      Volume = resvol
      DiskGroup = resdg
      )
```


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

export1 requires resvol
resvol requires resdg
resdg requires dr

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

ElifNone

Description	Monitors a file.	
Entry Points	Monitor— Checks if the specified file exists. If it does, the agent reports as offline. If it does not, the agent reports as online.	
Required Attribute	Type and Dimension	Definition
PathName	string-scalar	Specifies the complete pathname, starting with the slash (/) preceding the file name.

Type Definition

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

Sample Configuration

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


FileNone

Description	Monitors a file.	
Entry Points	Monitor— Checks if the specified file exists. If it does, the agent reports as online. If it does not, the agent reports as offline.	
Required Attribute	Type and Dimension	Definition
PathName	string-scalar	Specifies the complete pathname, starting with the slash (/) preceding the file name.

Type Definition

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

Sample Configuration

```
FileNone tmp_file01 (
    PathName = “/tmp/file01”
    )
```

FileOnOff Agent

Description	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 if the specified file exists. If it does, the agent reports as online. If it does not, the agent reports as offline.	
Required Attribute	**Type and Dimension**	**Definition**
PathName	string-scalar	Specifies the complete pathname, starting with the slash (/) preceding the file name.

Type Definition

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

Sample Configuration

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


FileOnOnly

Description	Creates and monitors files.	
Entry Points	◆ Online— Creates an empty file with the specified name, if one does not already exist. ◆ Monitor— Checks if the specified file exists. If it does, the agent reports as online. If it does not, the agent reports as offline.	
Required Attribute	**Type and Dimension**	**Definition**
PathName	string-scalar	Specifies the complete pathname, starting with the slash (/) preceding the file name.

Type Definition

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

Sample Configuration

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

IP Agent

Description	Manages the process of configuring an IP address on an interface.	
Entry Points	◆ Online—Checks if the IP address is in use by another system. Uses `ifconfig` to set the IP address on a unique alias on the interface. ◆ Offline—Brings down the IP address associated with the specified interface. Uses `ifconfig` to set the interface alias to `0.0.0.0` and the state to "down." ◆ Monitor—Monitors the interface to test if the IP address associated with the interface is alive.	
Required Attributes	**Type and Dimension**	**Definition**
Address	string-scalar	IP address associated with the interface.
Device	string-scalar	Name of the NIC resource associated with the IP address. Should only contain the resource name without an alias; for example, `le0`.
Optional Attributes	**Type and Dimension**	**Definition**
ArpDelay	integer-scalar	Number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about this IP address. Default is 1 second.
IfconfigTwice	integer-scalar	Causes an IP address to be configured twice, using an `ifconfig up-down-up` sequence. Increases probability of gratuitous `arps` (caused by `ifconfig up`) reaching clients. Default is 0.
NetMask	string-scalar	Netmask associated with the interface. The value of NetMask may be specified in decimal (base 10) or hexadecimal (base 16).
Options	string-scalar	Options for the `ifconfig` command.

Type Definition

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

Sample Configurations

Sample 1

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

Sample 2: NetMask in Decimal (base 10)

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

Sample 3: NetMask in Hexadecimal (base 16)

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

IPMultiNIC Agent

Description	Represents a virtual (logical) IP address configured as an alias on one interface of a MultiNICA resource. It also monitors the logical IP address. If the interface is faulted, the IPMultiNIC agent works with the MultiNICA resource to fail over to a backup interface. If multiple service groups have IPMultiNICs associated with the same MultiNICA resource, only one group will have the MultiNICA resource. The other groups will have Proxy resources pointing to it.	
Entry Points	◆ Online—Configures a virtual IP address on one interface of the MultiNICA resource. ◆ Offline—Removes a virtual IP address from one interface of the MultiNICA resource. ◆ Monitor—Checks if the virtual IP address is configured on one interface of the MultiNICA resource.	
Required Attributes	**Type and Dimension**	**Definition**
Address	string-scalar	Virtual IP address.
MultiNICResName	string-scalar	Name of associated MultiNICA resource.
Optional Attributes	**Type and Dimension**	**Definition**
ArpDelay	integer-scalar	Number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about this IP address. Default is 1 second.
IfconfigTwice	integer-scalar	Causes an IP address to be configured twice, using an `ifconfig up-down-up` sequence. Increases probability of gratuitous `arps` (caused by `ifconfig up`) reaching clients. Default is 0.
NetMask	string-scalar	NetMask for the virtual IP address. Default is “+”. The value of NetMask may be specified in decimal (base 10) or hexadecimal (base 16).
Options	string-scalar	The `ifconfig` options for the virtual IP address.

Type Definition

```
type IPMultiNIC (
   static str ArgList[] = { "MultiNICResName:Device", Address,
                                NetMask, "MultiNICResName:ArpDelay",
                                Options, "MultiNICResName:Probed",
                                MultiNICResName, ArpDelay,
                                IfconfigTwice }
   NameRule = IPMultiNIC_ + resource.Address
   str Address
   str NetMask
   str Options
   str MultiNICResName
   int ArpDelay = 1
   int IfconfigTwice = 0
   static int MonitorTimeOut = 120
)
```

Sample Configuration, IPMultiNIC and MultiNICA

For a detailed discussion of the following example, refer to “Sample Configuration: MultiNICA and IPMultiNIC” on page 29.

```
group grp1 (
   SystemList = { sysa, sysb }
   AutoStartList = { sysa }
   )
   MultiNICA mnic (
      Device@sysa = { le0 = "166.98.16.103",qfe3 = "166.98.16.103" }
      Device@sysb = { le0 = "166.98.16.104",qfe3 = "166.98.16.104" }
      NetMask = 255.255.255.0
      ArpDelay = 5
      Options = "trailers"
      RouteOptions@sysa = "default 166.98.16.103 0"
      RouteOptions@sysb = "default 166.98.16.104 0"
      )

   IPMultiNIC ip1 (
      Address = "166.98.14.78"
      NetMask = "255.255.255.0"
      MultiNICResName = mnic
      Options = "trailers"
      )

ip1 requires mnic
```

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

    IPMultiNIC ip2 (
        Address = "166.98.14.79"
        NetMask = "255.255.255.0"
        MultiNICResName = mnic
        Options = "trailers"
        )

    Proxy proxy (
        TargetResName = mnic
        )

ip2 requires proxy
```


Mount Agent

Description	Brings online, takes offline, and monitors a file system mount point.	
Entry Points	◆ Online—Mounts a block device on the directory. If mount fails, the agent tries to run `fsck` on the raw device to remount the block device. ◆ Offline—Unmounts the file system. ◆ Monitor—Determines if the file system is mounted. Checks mount status using the `stat` and `statvfs` commands.	
Required Attributes	**Type and Dimension**	**Definition**
BlockDevice	string-scalar	Block device for mount point.
MountPoint	string-scalar	Directory for mount point.
FSType	string-scalar	File system type, for example, vxfs or ufs.
Optional Attributes	**Type and Dimension**	**Definition**
FsckOpt	string-scalar	Options for `fsck` command.
MountOpt	string-scalar	Options for `mount` command.
SnapUmount	integer-scalar	If set to 1, causes VxFS snapshot mounts to be unmounted automatically when file system is unmounted. Default is 0 (No).

Type Definition

```
type Mount (
    str MountPoint
    str BlockDevice
    str FSType
    str MountOpt
    str FsckOpt
    int SnapUmount = 0
    NameRule = resource.MountPoint
    static str ArgList[] = { MountPoint, BlockDevice, FSType,
                                MountOpt, FsckOpt, SnapUmount }
)
```

Sample Configuration

```
Mount export1 (
    MountPoint= "/export1"
    BlockDevice = "/dev/dsk/c1t1d0s3"
    FSType = vxfs
    MountOpt = ro
    )
```


MultiNICA Agent

Description	Represents a set of network interfaces, and provides failover capabilities between them. Each interface in a MultiNICA resource has a base IP address, which can be the same or different. 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, it should not be associated with any other MultiNIC 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, and 2) Proxy resources that point to the MultiNIC resource in the other service groups.	
Entry Point	◆ Online—Not applicable. ◆ Offline—Not applicable. ◆ 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 interface.	
Required Attribute	**Type and Dimension**	**Definition**
Device	string-association	List of interfaces and their base IP addresses.
Optional Attributes	**Type and Dimension**	**Definition**
ArpDelay	integer-scalar	Number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about base IP address. Default is 1 second.
Handshake-Interval	integer-scalar	Helps compute the number of times the monitor will ping the network host after migrating to a new NIC. Default value is 90.
IfconfigTwice	integer-scalar	Causes an IP address to be configured twice, using an `ifconfig up-down-up` sequence. Increases probability of gratuitous `arps` (caused by `ifconfig up`) reaching clients. Default is 0.
NetMask	string-scalar	Netmask for the base IP address. Default is “+”. The value of NetMask may be specified in decimal (base 10) or hexadecimal (base 16).

NetworkHosts	string-vector	List of hosts on the network that will be pinged to determine if the network connection is alive. The IP address of the host should be entered instead of the HostName to prevent the monitor from timing out (DNS will cause the ping to hang). If this optional attribute is not specified, the monitor will test the NIC by pinging the broadcast address on the NIC. If more than one network host is listed, the monitor will return online if at least one of the hosts is alive (for example, `NetworkHosts = {“166.93.2.1”,“166.97.1.2”}`).
Options	string-scalar	The `ifconfig` options for the base IP address; for example, “trailers.”
PingOptimize	integer-scalar	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 each monitor cycle and detects the inactive interface within the cycle. Default is 1.
RouteOptions	string-scalar	String to add a route when configuring an interface. This string contains the information “destination gateway metric.” No routes are added if this string is set to `NULL`.

Notes about Using MultiNICA Agent

- If all the NICs configured in the `Device` attribute are down, the MultiNICA agent will fault the resource after a 2-3 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 during failover. (The engine log is located at `/var/VRTSvcs/log/engine_A.log`).
- The MultiNICA agent supports only one active NIC on one IP subnet; the agent will not work with multiple active NICs.
- The primary NIC must be configured before VCS is started. You can use the `ifconfig`(1M) command to configure it manually, or edit the file `/etc/hostname.`*<nic>* so that configuration of the NIC occurs automatically when the system boots. VCS plumbs and configures the backup NIC, so it does not require the file `/etc/hostname.`*<nic>*.

Type Definition

```
type MultiNICA (
   static str ArgList[] = { Device, NetMask, ArpDelay, Options,
                            RouteOptions, PingOptimize,
                            MonitorOnly, IfconfigTwice,
                            HandshakeInterval, NetworkHosts }
   static str Operations = None
   static int MonitorTimeout = 300
   NameRule = MultiNICA_ + group.Name
   str Device{}
   str NetMask
   int ArpDelay = 1
   str Options
   str RouteOptions
   int PingOptimize = 1
   int IfconfigTwice = 0
   int HandshakeInterval = 90
   str NetworkHosts[]
)
```

Sample Configuration: MultiNICA and IPMultiNIC

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

The MultiNICA resource fails over only the physical IP address to the backup NIC in the event of a failure. The logical IP addresses are configured by the IPMultiNIC agent. The resources `ip1` and `ip2`, shown in the following example, have an attribute called `Address`, which contains the logical IP address. In the event of a NIC failure on `sysa`, the physical IP address and the two logical IP addresses will fail over from `le0` to `qfe3`. In the event that `qfe3` fails, the address will fail back to `le0` if `le0` is reconnected.

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

If you have more than one group using the MultiNICA resource, the second group can use a Proxy resource to point to the MultiNICA resource in the first group. This prevents redundant monitoring of the NICs on the same system. The IPMultiNIC resource is always made dependent on the MultiNICA resource. See “IPMultiNIC Agent” on page 21.

```
group grp1 (
   SystemList = { sysa, sysb }
   AutoStartList = { sysa }
   )
   MultiNICA mnic (
      Device@sysa = { le0 = "166.98.16.103", qfe3 = "166.98.16.103" }
      Device@sysb = { le0 = "166.98.16.104", qfe3 = "166.98.16.104" }
      NetMask = "255.255.255.0"
      ArpDelay = 5
      Options = "trailers"
      RouteOptions@sysa = "default 166.98.16.103 0"
      RouteOptions@sysb = "default 166.98.16.104 0"
      )

   IPMultiNIC ip1 (
      Address = "166.98.14.78"
      NetMask = "255.255.255.0"
      MultiNICResName = mnic
      Options = "trailers"
      )
```


```
ip1 requires mnic


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

      IPMultiNIC ip2 (
         Address = "166.98.14.79"
         NetMask = "255.255.255.0"
         MultiNICResName = mnic
         Options = "trailers"
         )
      Proxy proxy (
         TargetResName = mnic
         )

ip2 requires proxy
```

NFS Agent

Description	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're not, it starts them and exits. ◆ Offline—Not applicable. ◆ 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 calls succeed, the resource is reported as online.	
Optional Attribute	**Type and Dimension**	**Definition**
Nservers	integer-scalar	Specifies the number of concurrent NFS requests the server can handle. Default is 16.

Type Definition

```
type NFS (
   int Nservers = 16
   NameRule = "NFS_" + group.Name + "_" + resource.Nservers
   static str ArgList[] = { Nservers }
   static str Operations = OnOnly
   static int RestartLimit = 1
)
```

Sample Configuration

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


NIC Agent

Description	Monitors the configured NIC. If a network link fails, or if there is a problem with the device card, the resource is marked as offline. The NIC listed in the `Device` attribute must have an administration IP address, which is the default IP address assigned to the physical interface of a host on a network. The NIC agent will not configure network routes or an administration IP address.	
Entry Points	◆ Online—Not applicable. ◆ Offline—Not applicable. ◆ Monitor—Tests the network card and network link. Uses the DLPI (Data Link Provider Interface) layer to send and receive messages across the driver. Pings the 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 as offline.	
Required Attribute	**Type and Dimension**	**Definition**
Device	string-scalar	NIC name.
Optional Attributes	**Type and Dimension**	**Definition**
PingOptimize	integer-scalar	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. Default is 1.
NetworkHosts	string-vector	List of hosts on the network that will be pinged to determine if the network connection is alive. The IP address of the host should be entered instead of the HostName to prevent the monitor from timing out (DNS problems can cause the ping to hang); for example, `166.96.15.22`. If this optional attribute is not specified, the monitor will test the NIC by pinging the broadcast address on the NIC. If more than one network host is listed, the monitor will return online if at least one of the hosts is alive.
NetworkType	string-scalar	Type of network, such as Ethernet (ether), FDDI (fddi), Token Ring (token), etc.

Type Definition

```
type NIC (
    str Device
    str NetworkType
    str NetworkHosts[]
    NameRule = group.Name + "_" + resource.Device
    int PingOptimize = 1
    static str ArgList[] = { Device, NetworkType, NetworkHosts,
                             PingOptimize }
    static str Operations = None
)
```

Sample Configurations

Sample 1: Without Network Hosts, Using the Default Ping Mechanism

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

Sample 2: With Network Hosts

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


Phantom Agent

Description	Enables groups with no OnOff resources to display the correct status. **Note** Include Phantom resource types in parallel groups only. Service groups that do not include OnOff resources as members are not brought online, even if their member resources are brought online, because the status of the None and OnOnly resources are not considered when deciding if a group is online.
Entry Point	◆ Online—Not applicable. ◆ Offline—Not applicable. ◆ Monitor—Determines status based on the status of its group.

Type Definition

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

Sample Configurations

Sample 1

```
Phantom (
   )
```

Sample 2

Example 2 shows a complete `main.cf`, in which the FileNone agent and the Phantom agent are in the same group. Depending on the status reported by FileNone (which has no OnOff resources), the Phantom agent onlines or offlines the group.

```
include "types.cf"

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

system sysa

system sysb

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

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

        FileNone my_file_none (PathName = "/tmp/file_none"
                )
        Phantom my_phantom (
                )

        // resource dependency tree
        //
        //      group maingroup
        //      {
        //      Phantom my_Phantom
        //      FileNone my_file_none
        //      }
```


Process Agent

Description	Starts, stops, and monitors a process specified by the user.	
Entry Points	◆ Online—Starts the process with optional arguments. ◆ Offline—VCS sends a `SIGTERM`. If the process does not exit within one second, VCS sends a `SIGKILL`. ◆ Monitor—Checks to see if the process is alive by scanning the process table for the name of the executable pathname and argument list.	
Required Attribute	**Type and Dimension**	**Definition**
PathName	string-scalar	Defines complete pathname for accessing an executable program, including the program name. Pathname must not exceed 80 characters.
Optional Attribute	**Type and Dimension**	**Definition**
Arguments	string-scalar	Passes arguments to the process. Pathname must not exceed 80 characters. **Note** Multiple tokens must be separated by one space only. String cannot accommodate more than one space between tokens, or leading or trailing whitespace characters. Arguments must not exceed 80 characters total

Type Definition

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

Sample Configurations

Sample 1

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

Sample 2

```
include "types.cf"

cluster ProcessCluster (
      UserNames = { admin = cDXedEFSN4P56 }
      Factor = { runque = 5, memory = 1, disk = 10, cpu = 25,
                 network = 5 }
      MaxFactor = { runque = 100, memory = 10, disk = 100,
                 cpu = 100, network = 100 }
      )

system sysa

system sysb

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

group 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
         //      }
```


Proxy Agent

Description	Mirrors the state of another resource on the local or a remote system.	
Entry Point	◆ Online—Not applicable. ◆ Offline—Not applicable. ◆ Monitor—Determines status based on the target resource status.	
Required Attribute	**Type and Dimension**	**Definition**
TargetResName	string-scalar	Name of the target resource whose status is mirrored by Proxy resource. NOTE: The target resource must be in a different resource group from the Proxy resource.
Optional Attribute	**Type and Dimension**	**Definition**
TargetSysName	string-scalar	Mirror the status of the `TargetResName` on system specified by the TargetSysName variable. If attribute is not specified, Proxy resource assumes the system is local.

The proxy agent mirrors the status of a resource (the TargetResName) in another resource group. The agent provides a means to specify and modify one resource and have it reflected by its proxies.

Type Definition

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

Sample Configurations

Sample 1

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

Proxy (
      TargetResName = "tmp_VRTSvcs_file1"
      )
```

Sample 2

```
// Proxy resource to mirror the state of the resource
// tmp_VRTSvcs_file1 on sys1.

Proxy (
      TargetResName = "tmp_VRTSvcs_file1"
      TargetSysName = "sys1"
      )
```

Sample 3

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

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

   MultiNICA mnic (
      Device@sysa = { le0 = "166.98.16.103",qfe3 = "166.98.16.103" }
      Device@sysb = { le0 = "166.98.16.104",qfe3 = "166.98.16.104" }
      NetMask = "255.255.255.0"
      ArpDelay = 5
      Options = "trailers"
      RouteOptions@sysa = "default 166.98.16.103 0"
      RouteOptions@sysb = "default 166.98.16.104 0"
      )
   IPMultiNIC ip1 (
      Address = "166.98.14.78"
      NetMask = "255.255.255.0"
```


```
        MultiNICResName = mnic
        Options = "trailers"
        )

ip1 requires mnic


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

ServiceGroupHB Agent

Description	The agent starts, stops, and monitors disk-based heartbeats associated with service groups. (See the *VERITAS Cluster Server 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 the Disks attribute). The local system, via the ServiceGroupHB agent, attempts 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 onlines and monitors the resource. If disk ownership falls below the AllOrNone requirement, VCS attempts to fail over the group to another node.	
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.	
Required Attribute	**Type and Dimension**	**Definition**
Disks	string-vector	Specifies, in paired values, the disk partition and the starting block location to use for the heartbeat. Note that a block device partition is used for the disk heartbeating; for example, `/dev/dsk/c1t2d0s3`. A block device partition containing one or more heartbeat regions cannot be used for any other purpose. If the same partition is used for more than one heartbeat region, starting block numbers must be at least 64K (128 disk blocks) apart.
AllOrNone	boolean-scalar	Specifies number of disks for which "ownership" is required to Online the resource: ◆ all available disks (AllOrNone = 1) ◆ a simple majority of available disks (AllOrNone = 0) The default value is 1.

Type Definition

```
type ServiceGroupHB (
   static str ArgList[] = { Disks, AllOrNone }
   NameRule = SGHB_ + resource.Disks
   str Disks[]
   boolean AllOrNone = 1
)
```

Sample Configuration

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

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

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

        ServiceGroupHB sghb1 (
                Disks = { c1t2d0s4, 64, c2t2d0s4, 64, c3t2d0s4, 64 }
                AllorNone = 0
                )

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

group groupy (
.
.
        )
```

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

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

exp1 requires sghb1
exp2 requires sghb2

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


Share Agent

Description	Shares, unshares, and monitors a single local resource for exporting an NFS file system to be mounted by remote systems.	
Entry Points	◆ Online—Shares an NFS file system. ◆ Offline—Unshares an NFS file system. ◆ Monitor—Reads `/etc/dfs/sharetab` file and looks for an entry for the file system specified by `PathName`. If the entry exists, it returns as online.	
Required Attribute	**Type and Dimension**	**Definition**
PathName	string-scalar	Pathname of the file system to be shared.
Optional Attributes	**Type and Dimension**	**Definition**
Options	string-scalar	Options for the `share` command.

Type Definition

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

Sample Configuration

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

Volume Agent

Description	Brings online, takes offline, and monitors a VERITAS Volume Manager volume.	
Entry Points	◆ Online—Using the `vxvol` command, this agent starts the volume. ◆ Offline—Using the `vxvol` command, this agent stops the volume. ◆ Monitor—Determines if the volume is online or offline by reading a block from the raw device interface to the volume.	
Required Attributes	**Type and Dimension**	**Definition**
DiskGroup	string-scalar	Disk group name.
Volume	string-scalar	Volume name.

Type Definition

```
type Volume (
   str Volume
   str DiskGroup
   NameRule = resource.DiskGroup + "_" + resource.Volume
   static str ArgList[] = { Volume, DiskGroup }
)
```

Sample Configuration

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


Index