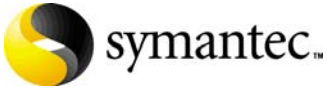
Veritas™ Cluster Server Bundled Agents Reference Guide

HP-UX 11i v3

5.0.1



Veritas Cluster Server Bundled Agents Reference Guide

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

Product version: 5.0.1

Document version: 5.0.1.0

Legal Notice

Symantec Corporation
350 Ellis Street
Mountain View, CA 94043
http://www.symantec.com

Technical Support

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

Symantec's maintenance offerings include the following:

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

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

www.symantec.com/techsupp

Contacting Technical Support

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

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

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

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

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

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

Licensing and registration

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

www.symantec.com/techsupp

Customer service

Customer service information is available at the following URL:

www.symantec.com/techsupp

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

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

Documentation feedback

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

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

are reporting.

Maintenance agreement resources

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

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

Additional enterprise services

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

Enterprise services that are available include the following:

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

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

www.symantec.com

Select your country or language from the site index.

Contents

Chapter 3 Network agents

Chapter 6 Infrastructure and support agents

Chapter 7 Testing agents

Chapter 1

Introduction

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

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

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

Agents can:

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

For a more detailed overview of how agents work, refer to the *Veritas Cluster Server User's Guide*.

Resources and their attributes

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

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

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

Modifying agents and their resources

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

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

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

Attributes

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

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

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

Table 1-1 Attribute data types

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

Table 1-1 Attribute data types

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

Table 1-2 Attribute dimensions

Dimension	Description
scalar	A scalar has only one value. This is the default dimension.
vector	A vector is an ordered list of values. Each value is indexed using a positive integer beginning with zero. A set of brackets ([]) denotes that the dimension is a vector. Find the specified brackets after the attribute name on the attribute definition in the types.cf file.
keylist	A keylist is an unordered list of unique strings.
association	An association is an unordered list of name-value pairs. An equal sign separates each pair. A set of braces ({}) denotes that an attribute is an association. Braces are specified after the attribute name on the attribute definition in the types.cf file, for example: str SnmpConsoles{}.

Chapter 2

Storage agents

This chapter contains:

About the storage agents

Use storage agents to Monitor shared storage.

DiskGroup agent

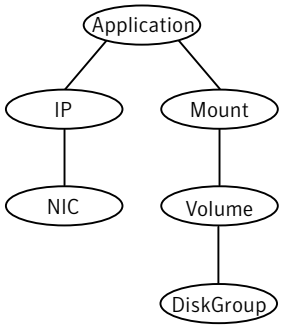
The DiskGroup agent brings online, takes offline, and monitors Veritas Volume Manager (VxVM) disk groups. This agent uses VxVM commands. You can use this agent to monitor or make disk groups highly available.

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

Dependencies

The DiskGroup resource has no required resources.

Figure 2-1 Sample service group that includes a DiskGroup resource



Agent functions

Online	Imports the disk group using the `vxdg` command.
Offline	Deports the disk group using the `vxdg` command.
Monitor	Determines if the disk group is online or offline using the `vxdg` command. The Monitor function changes the value of the VxVM noautoimport flag from off to on. This action allows VCS to maintain control of importing the disk group. The following command changes the autoimport flag back to on: `# vxdg -g disk_group set autoimport=yes`
Clean	Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.

Info

The DiskGroup info agent function gets information from the Volume Manager and displays the type and free size for the DiskGroup resource.

Initiate the info agent function by setting the InfoInterval timing to a value greater than 0.

In the following example, the info agent function executes every 60 seconds:

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

The command to retrieve information about the DiskType and FreeSize of the DiskGroup resource is:

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

Output includes:

```
DiskType auto:cdsdisk
FreeSize 12765712
```

Action

Different action agent functions follow:

- license.vfd
  Checks for valid Veritas Volume manager license–if one is not found use the vxlicinst utility to install a valid license key.
- disk.vfd
  Checks if all disks in diskgroup are visible on host—if it fails, check if the path to disks exists from the host and check if LUN masking and zoning are set properly.
- udid.vfd
  Checks the UDIDs (unique disk identifiers) of disks on the cluster nodes—if it fails, ensure that the disks that are used for the disk group are the same on all cluster nodes.
- verifyplex.vfd
  Checks if the number of plexes on each site for the Campus Cluster setup are set properly—if it fails, check that the sites, disks, and plexes are set properly for a Campus Cluster setup.
- volinuse
  Checks if open volumes are in use or file systems on volumes that are mounted outside of VCS configuration.

See "High availability fire drill" on page 29.

State definitions

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

Attributes

Table 2-1 Required attributes

Required attribute	Description
DiskGroup	Name of the disk group that is configured with Veritas Volume Manager. Type and dimension: string-scalar Example: "diskgroup1"

Table 2-2 Optional attributes

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

Table 2-2 Optional attributes

Optional attributes	Description
PanicSystemOnDGLoss	Determines whether to panic the node if the disk group becomes disabled. A loss of storage connectivity can cause the disk group to become disabled. If the value of this attribute is 1 and the disk group becomes disabled, the node panics. If PanicSystemOnDGLoss is set to 1, and the Monitor agent function (entry point) hangs a consecutive number of times per the value of the FaultOnMonitorTimeouts attribute, then the node panics. **Note:** System administrators may want to set a high value for FaultOnMonitorTimeouts to increase system tolerance. If the value of the attribute is 0 and the disk group becomes disabled, the following occurs: ■ If the cluster has I/O fencing enabled, the DiskGroup resource is marked FAULTED. This state results in the agent attempting to take the service group offline. As part of bringing the DiskGroup resource offline, the agent attempts to deport the disabled disk group. Even if disabled disk group fails to deport, the DiskGroup resource enters a FAULTED state. This state enables the failover of the service group that contains the resource. To fail back the DiskGroup resource, manually deport the disk group after restoring storage connectivity. ■ If the cluster does not use I/O fencing, a message is logged and the resource is reported ONLINE. The resource is reported ONLINE so that it does not fail over, which ensures data integrity. **Note:** The PanicSystemOnDGLoss attribute does not depend on the MonitorReservation attribute. Type and dimension: boolean-scalar Default: 0
StartVolumes	If value is 1, the DiskGroup online function starts all volumes belonging to that disk group after importing the group. Type and dimension: string-scalar Default: 1

Table 2-2 Optional attributes

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

Resource type definition

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

DiskGroup agent notes

The DiskGroup agent has the following notes:

- "High availability fire drill" on page 29

High availability fire drill

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

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

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

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

Sample configurations

DiskGroup resource configuration

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

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

DiskGroup, Volume, and Mount dependencies configuration

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
```

DiskGroupSnap agent

Use the DiskGroupSnap agent to perform fire drills in a campus cluster. The DiskGroupSnap agent enables you to verify the configuration and data integrity in a Campus Cluster environment with VxVM stretch mirroring. The agent also supports SCSI-3 fencing.

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

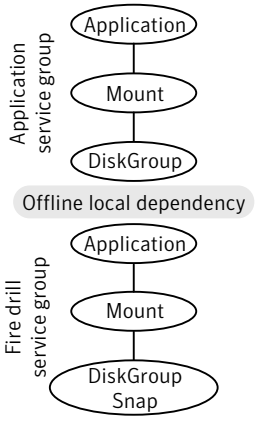
For important information about this agent, refer to:

"DiskGroupSnap agent notes" on page 33

Dependencies

The DiskGroupSnap resource does not depend on any other resources. The service group that contains the DiskGroupSnap agent has an offline local dependency on the application's service group. The offline local dependency is to make sure the firedrill service group and the application service group are not online at the same site at the same time.

Figure 2-2 Sample service group that includes a DiskGroupSnap resource



Agent functions

Online	Verifies that the application's disk group is in a valid campus cluster configuration. It detaches the site that the value of the FDSiteName attribute specifies. It then creates another disk group to be used for the fire drill on the detached site.
Offline	This re-attaches the site that the value of the FDSiteName attribute specifies back to the application's disk group.
Monitor	Monitors the DiskGroupSnap resource.
Clean	Takes the DiskGroupSnap resource offline.
Open	If the DiskGroupSnap resource has a parent resource that is not ONLINE, then it deletes the online lock file of the DiskGroupSnap resource. This marks the DiskGroupSnap resource as OFFLINE.

State definitions

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

Attributes

Table 2-3 Required attributes

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

Table 2-3 Required attributes

Required attribute	Description

FDSiteName	At a site, this is the unique VxVM site name tag for the fire drill disks. You can run the fire drill in the following configurations:
	■ In the Gold configuration, a site has a dedicated set of fire drill disks. In Figure 2-4, the disaster recovery site uses a Gold configuration.
	■ In the Bronze configuration, a site uses its data disks as fire drill disks. In Figure 2-4, the primary site uses a Bronze configuration.
	Type and dimension: string-scalar
	Example:
	The value for the FDSiteName attribute for the configuration for Figure 2-4 is:
	`"FDSiteName@Node_A = pri"`
	`"FDSiteName@Node_B = pri"`
	`"FDSiteName@Node_C = dr_fd"`
	`"FDSiteName@Node_D = dr_fd"`

DiskGroupSnap agent notes

The DiskGroupSnap agent has the following notes:

Configuring the SystemZones attribute for the fire drill service group

You must assign the local system values to the SystemZones attribute of the application's service group. You set these values so that the service group fails over in the same zone before it tries to fail over across zones. For more information about campus cluster setup, refer to the *Veritas Cluster Server User's Guide*.

For example, you set up the service group's SystemZones attribute for two zones: 0 and 1. You want the service group on Node_A and Node_B to fail over between the two nodes before it comes up elsewhere. The application and its fire drill service group both have the following values for the SystemZones attribute:

```
SystemZones = { Node_A = 0, Node_B = 0, Node_C = 1, Node_D = 1 }
```

Configuring the firedrill service group

In the DiskGroupSnap agent, the application-level resources (for example process resources, application resources, or Oracle resources, etc.) can have the same attribute values in the firedrill service group and the application service group. The reuse of the same values for the attributes can result in VCS reporting the wrong resources as online.

Set the FireDrill type-level attribute to 1 for those types. For example, if the Oracle and Listener resources are configured identically, set the FireDrill attribute for Oracle and Listener to 1:

```
haconf –makerw
hatype –modify Oracle FireDrill 1
hatype –modify Listener FireDrill 1
haconf -dump –makero
```

Adding the ReuseMntPt attribute to the ArgList attribute for the Mount agent type

If you plan to use a Mount resource in a firedrill service group, you must add the ReuseMntPt attribute to ArgList and set its value to 1.

To add the ReuseMntPt attribute to the ArgList attribute and set its value to 1

1. Make the configuration read and write.

   # **haconf -makerw**

2. Add the ReuseMntPt attribute to the ArgList attribute.

   # **hatype -modify Mount ArgList -add ReuseMntPt**

3. Change the value of the ReuseMntPt attribute to 1 for the firedrill's Mount resource.

   # **hares -modify *firedrill_mount_resource_name* ReuseMntPt 1**

4. Change the value of the of the ReuseMntPt attribute to 1 for the original Mount resource.

   # ***hares -modify original_mount_resource_name ReuseMntPt 1***

5. Make the configuration read only.

   # **haconf –dump –makero**

Configuration considerations

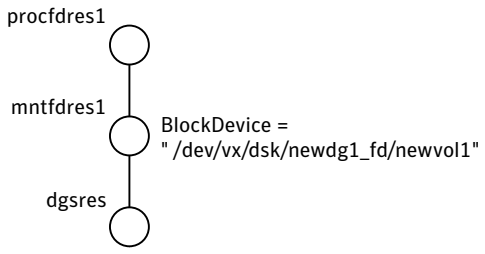
Keep the following recommendations in mind:

- Do not bring the DiskGroupSnap resource online in the SystemZone where the application service group is online.
- Make sure that the fire drill service group and the application service group both use the same values for the SystemZones attribute.
- Do not use Volume resources in the fire drill service group. The DiskGroupSnap agent internally uses the `vxvol` command to start all the volumes in the fire drill disk group.
- In large setups, you may need to tweak the various timer values so that the timers do not timeout while waiting for VxVM commands to complete. The timers you need to tweak are the OfflineTimeout for DiskGroupSnap resource and the MonitorInterval and ActionTimeout for the associated DiskGroup resource, for example:

```
haconf –makerw
hares -override DGSres OfflineTimeout
hares -modify DGSres OfflineTimeout 600
hares -override DGSres MonitorInterval
hares -modify DGSres MonitorInterval 1200 (this has to be twice
the value intended for ActionTimeout below)
hares -override DGSres ActionTimeout
hares -modify DGSres ActionTimeout 600
haconf –dump –makero
```

- When you create the fire drill service group, in general use the same attribute values that you use in the application service group.
  The BlockDevice attribute of the Mount resource changes between the application service group and the fire drill service group. In the BlockDevice path, you must append an _fd to the disk group name portion, for example, /dev/vx/dsk/newdg1/newvol1 becomes /dev/vx/dsk/newdg1_fd/newvol1.
  Figure 2-3 shows the changes to resource values for the fire drill service group; note that the Volume resource is not included.

Figure 2-3 Sample resource values for a DiskGroupSnap resource



Agent limitations

The following limitations apply to the DiskGroupSnap agent:

- The DiskGroupSnap agent does not support Volume Sets.
- Do not use the DiskGroupSnap agent in a Storage Foundation RAC environment.
- The online and offline operations of the DiskGroupSnap resource invokes VCS action entry points to run VxVM commands to detach/reattach the fire drill site. Since VxVM requires that these commands are run on the node where the disk group is imported, the disk group has to be imported on some node in the cluster before these operations.
- Take the firedrill service group offline before you shut down VCS on any node. If you fail to take the firedrill service group offline before you shut down VCS, you must manually reattach the firedrill site to the disk group to continue to perform fire drills.
- Use the enclosures that have the ASL/APM libraries that are supported in the Veritas Volume Manager. To view the supported enclosures, use the `vxddladm listsupport` command.

Resource type definition

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

Sample configurations

In Figure 2-4, the Primary site is in the Bronze configuration and the Disaster recovery site is in a Gold configuration.

Since the Primary site does not have dedicated fire drill disks, it is in a Bronze configuration. In the Bronze configuration, you re-purpose the mirror disks in the disaster recovery site to serve as fire drill test disks. The drawback with the Bronze configuration is that if a disk failure occurs when the fire drill is online at the Primary, it results in a site failure.

The FDSiteName value in a bronze configuration is the VXVM site name. For this configuration, the FDSiteName attribute values for the nodes at the primary follow:

```
FDSiteName@Node_A = pri
FDSiteName@Node_B = pri
```

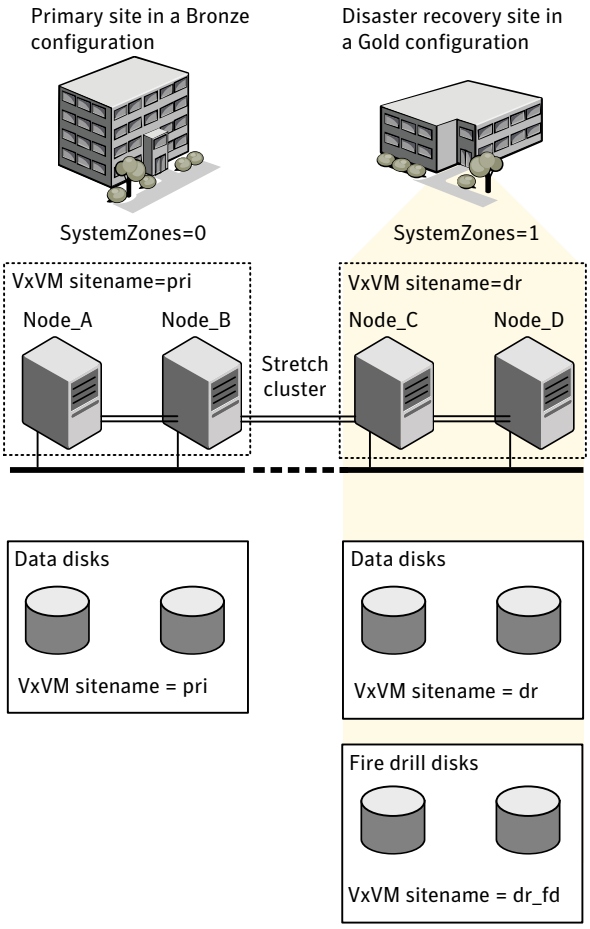
The Disaster Recovery site is in a Gold configuration as it has dedicated fire drill disks at the site. For the FDSiteName attribute, use the VxVM site tag given to the firedrill disks. For this configuration, the FDSiteName attribute values for the nodes at the Disaster recovery site follow:

```
FDSiteName@Node_C = dr_fd
FDSiteName@Node_D = dr_fd
```

Set values for the SystemZones attribute to zero for Node_A and Node_B, and one for Node_C and Node_D. For example:

```
SystemZones = { Node_A = 0, Node_B = 0, Node_C = 1, Node_D = 1 }
```

Figure 2-4 Primary site with the Bronze configuration and the disaster recovery site with the Gold configuration



Typical main.cf configuration

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

```
group dgfdsg (
    SystemList = { Node_A = 0, Node_B = 1, Node_C = 2, Node_D = 3 }
    SystemZones = { Node_A = 0, Node_B = 0, Node_C = 1, Node_D = 1}
    )

    DiskGroupSnap dgsres (
        TargetResName = dgres
```

```
        FDSiteName @Node_A = pri
        FDSiteName @Node_B = pri
        FDSiteName @Node_D = dr_fd
        FDSiteName @Node_D = dr_fd
        )

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

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

    Process procfdres1 (
        PathName = "/usr/bin/ksh"
        Arguments = "/scrib.sh /dgsfs1"
        )

    Process procfdres2 (
        PathName = "/usr/bin/ksh"
        Arguments = "/scrib.sh /dgsfs2"
        )

    requires group dgsg offline local
    mntfdres1 requires dgfdres
    mntfdres2 requires dgfdres
    procfdres1 requires mntfdres1
    procfdres2 requires mntfdres2
```

The application's service group (the actual service group) follows:

```
group dgsg (
    SystemList = { Node_A = 0, Node_B = 1, Node_C = 2, Node_D = 3 }
    SystemZones = { Node_A = 0, Node_B = 0, Node_C = 1, Node_D = 1}
    )

    DiskGroup dgres (
        DiskGroup = newdg1
        )

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

```
    Mount mntres2 (
        MountPoint = "/dgsfs2"
        BlockDevice = "/dev/vx/dsk/newdg1/newvol2"
        FSType = vxfs
        FsckOpt = "-y"
        )

    Process procres1 (
        PathName = "/usr/bin/ksh"
        Arguments = "/scrib.sh /dgsfs1"
        )

    Process procres2 (
        PathName = "/usr/bin/ksh"
        Arguments = "/scrib.sh /dgsfs2"
        )

    mntres1 requires dgres
    mntres2 requires dgres
    procres1 requires mntres1
    procres2 requires mntres2
```

Oracle main.cf configuration

The following Oracle configuration has been simplified for presentation within this guide.

```
group fd_oragrp (
        SystemList = { Node_A = 0, Node_B = 1 }
        AutoStart = 0
        SystemZones = { Node_A = 0, Node_B = 1 }
        )

        DiskGroupSnap dgres (
                FDSiteName @Node_A = siteA
                FDSiteName @Node_B = siteB
                TargetResName = oradg_res
                )

        IP fd_oraip (
                Device = e1000g0
                Address = "10.198.95.191"
                )

        Mount fd_archmnt (
                FsckOpt = "-y"
                ReuseMntPt = 1
                BlockDevice = "/dev/vx/dsk/oradg_fd/archive_vol"
                MountPoint = "/ora_archive"
                FSType = vxfs
```

```
                )

        Mount fd_datamnt (
                FsckOpt = "-y"
                ReuseMntPt = 1
                BlockDevice = "/dev/vx/dsk/oradg_fd/data_vol"
                MountPoint = "/ora_data"
                FSType = vxfs
                )

        NIC fd_oranic (
                Device = e1000g0
                )

        Netlsnr fd_LSNR (
                Home = "/opt/oracle/ora_home"
                Owner = oracle
                )

        Oracle fd_Ora_01 (
                Owner = oracle
                Home = "/opt/oracle/ora_home"
                Sid = Ora_01
                )

group oragrp (
        SystemList = { Node_A = 0, Node_B = 1 }
        AutoStartList = { Node_A, Node_B }
        SystemZones = { Node_A = 0, Node_B = 1 }
        )

        DiskGroup oradg_res (
                DiskGroup = oradg
                )

        IP Node_A4-vip (
                Device = e1000g0
                Address = "10.198.95.192"
                )

        Mount arch_mnt (
                FsckOpt = "-y"
                ReuseMntPt = 1
                BlockDevice = "/dev/vx/dsk/oradg/archive_vol"
                MountPoint = "/ora_archive"
                FSType = vxfs
                )

        Mount data_mnt (
                FsckOpt = "-y"
                ReuseMntPt = 1
                BlockDevice = "/dev/vx/dsk/oradg/data_vol"
```

```
        MountPoint = "/ora_data"
        FSType = vxfs
        )

NIC nic_Node_A4vip (
        Device = e1000g0
        )

Netlsnr LSNR (
        Home = "/opt/oracle/ora_home"
        Owner = oracle
        )

Oracle Ora_01 (
        Owner = oracle
        Home = "/opt/oracle/ora_home"
        Sid = Ora_01
        )

Volume arch_vol (
        Volume = archive_vol
        DiskGroup = oradg
        )

Volume data_vol (
        Volume = data_vol
        DiskGroup = oradg
        )
```

Volume agent

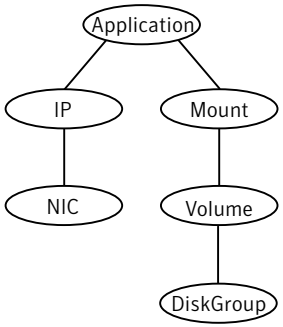
The Volume agent brings online, takes offline, and monitors a Veritas Volume Manager (VxVM) volume. Use the agent to make a volume highly available.

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

Dependencies

Volume resources depend on DiskGroup resources.

Figure 2-5 Sample service group that include a Volume resource



Agent functions

Online	Uses the `vxrecover` command to start the volume.
Offline	Uses the `vxvol` command to stop the volume.
Monitor	Attempts to read a block from the raw device interface to the volume to determine if the volume is online, offline, or unknown.
Clean	Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.

State definitions

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

Attributes

Table 2-4 Required attributes

Required attribute	Description
DiskGroup	Name of the disk group that contains the volume. Type and dimension: string-scalar Example: `"DG1"`
Volume	Name of the volume from disk group specified in DiskGroup attribute. Type and dimension: string-scalar Example: "DG1Vol1"

Table 2-5 Internal attribute

Optional attribute	Description
NumThreads	Number of threads used within the agent process for managing resources. This number does not include threads used for other internal purposes. Do not modify this attribute. Setting this attribute to a higher value may result in agent function timeouts due to serialization of underlying commands. Default: 1

Resource type definition

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

Sample configuration

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

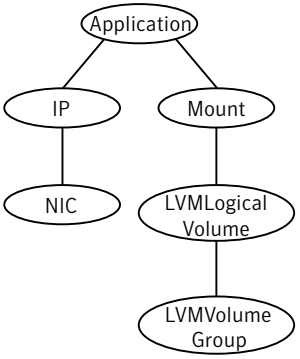
LVMLogicalVolume agent

The LVMLogicalVolume agent brings online, takes offline, and monitors Logical Volume Manager (LVM) logical volumes. You can use this agent to make volume groups and logical volumes highly available and to monitor them.

Dependencies

LVMLogicalVolume resources depend on LVMVolumeGroup resources.

Figure 2-6 Sample service group that includes a LVMLogicalVolume resource



Agent functions

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.

Attributes

Table 2-6 Required attributes

Required attribute	Description
LogicalVolume	Name of the logical volume. Type and dimension: string-scalar Example: "1vol1"
VolumeGroup	Name of the volume group containing the logical volume. Type and dimension: string-scalar Example: "vg1"

Table 2-7 Optional attributes

Optional attribute	Description
VolumeIOTimeout	The time for which the agent should wait before it returns an OFFLINE state when IO to the volume hangs. Default: 15 Minimum value: 3 seconds Maximum value: No maximum value, but the higher the value the higher the failover time required.

Resource type definition

```
type LVMLogicalVolume (
    static int NumThreads = 1
    static str ArgList[] = { LogicalVolume, VolumeGroup,
    VolumeIOTimeout }
    str LogicalVolume
    str VolumeGroup
    int VolumeIOTimeout = 15
)
```

Sample configuration

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

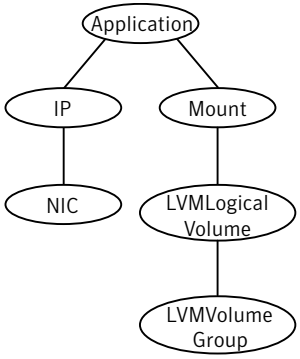
LVMVolumeGroup agent

The LVMVolumeGroup agent activates, deactivates, and monitors LVM volume groups. You can use this agent to make volume groups and logical volumes highly available and to monitor them.

Dependencies

The LVMVolumeGroup resource has no dependencies.

Figure 2-7 Sample service group that includes a LVMVolumeGroup resource



Agent functions

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 SMH to import a volume group.
Offline	Deactivates a volume group with the `vgchange` command.
Monitor	Determines whether the volume group is available.
Clean	Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

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.

Attributes

Table 2-8 Required attributes

Required attribute	Description
VolumeGroup	The name of the volume group that is configured with Logical Volume Manager. Type and dimension: string-scalar Example: "sharevg"

Resource type definition

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

Sample configurations

Configuration 1

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

Configuration 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
```

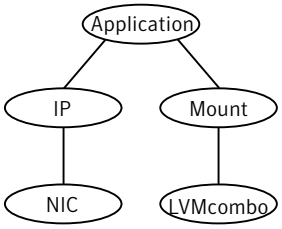
LVMCombo agent

The LVMCombo agent controls the activation and deactivation of the logical volumes and the Logical Volume group. You can use this agent to make volume groups and logical volumes highly available.

Dependencies

No dependencies exist for the LVMCombo resource.

Figure 2-8 Sample service group that includes an LVMcombo resource



Agent functions

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 should not activate it. This agent does not import volume groups because of the way LVM stores configuration information. Use the HP-UX SMH 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 activated, the resource is online. Otherwise, the resource is reported offline.

Note: The monitor agent function 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 situation 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.

Attributes

Table 2-9 Required attributes

Required Attribute	Description
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"

Table 2-10 Optional attributes

Optional Attribute	Description
VolumeIOTimeout	The time for which the agent waits before it returns an OFFLINE state when I/Os to the volume hangs. Default: "15" Minimum value: 3 seconds Maximum value: No maximum value, but the higher the value the higher the failover time required.

Resource type definition

```
type LVMCombo (
    static keylist SupportedActions = { volinuse }
    static str ArgList[] = { VolumeGroup, LogicalVolumes,
    VolumeIOTimeout }
    str VolumeGroup
    str LogicalVolumes[]
    int VolumeIOTimeout = 15
)
```

Physical volumes associated with volume groups

For all the Physical Volumes (PV) that are associated with a Volume Group, set the timeout to a smaller value than specified in the VolumeIOTimeout attribute of the resource. For example, if you specify an IOTimeout to equal 15 seconds, update the PV Timeout to a value that is less than 15 seconds.

Use the following command to change the timeout:

```
# pvchange -t time /dev/dsk/PV Used
# pvchange -t time Physical Volume
```

For example:

```
# pvchange -t 10 /dev/dsk/c2t4d4
```

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
```

Mount agent

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

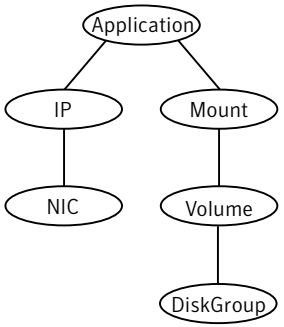
For important information about this agent, refer to:

"Mount agent notes" on page 63

Dependencies

No dependencies exist for the Mount resource.

Figure 2-9 Sample service group that includes a Mount resource



Agent functions

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

Info

The Mount info agent function executes the command:

```
bdf mount_point
```

The output displays Mount resource information:

```
Size Used Avail Use%
```

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

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

The command to retrieve information about the Mount resource is:

```
hares -value mountres ResourceInfo
```

Output includes:

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

Action

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

attr_changed

Unlocks the mounts when you change the value of the VxFSMountLock attribute from 1 to 0 and vice-versa.

State definitions

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

Attributes

Table 2-11 Required attributes

Required attribute	Description

BlockDevice	Block device for mount point.
	For LVM2, use the actual path to the volume.
	Type and dimension: string-scalar
	Examples:
	■ VxVM example "/dev/vx/dsk/myvcs_dg/myvol"
	■ LVM2 example
	`# ls -la /dev/ora_vg/ora_vol`
	`lrwxrwxrwx  1 root root 26 Apr 17 04:48 /dev/ora_vg/ora_vol -> /dev/mapper/ora_vg-ora_vol`
	Use the path /dev/mapper/ora_vg-ora_vol for the BlockDevice attribute.

FsckOpt	Mandatory for non-NFS mounts. Use this attribute to specify options for the `fsck` command. You must correctly set this attribute for local mounts. If the mount process fails, the `fsck` command is executed with the specified options before it attempts to remount the block device. Its value must include either `-y` or `-n`. Refer to the fsck manual page for more information. The `-y` argument enables the VxFS file systems to perform a log replay before a full `fsck` operation. For NFS mounts, the value of this attribute is not applicable and is ignored. Type and dimension: string-scalar VxFS example: `-y`
FSType	Type of file system. Supports vxfs, hfs, or nfs. Type and dimension: string-scalar Example: "nfs"

Table 2-11 Required attributes

Required attribute	Description
MountPoint	Directory for mount point. Type and dimension: string-scalar Example: "/campus1"

Table 2-11 Required attributes

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

Table 2-12 Optional attributes

Optional attribute	Description
CkptUmount	If the value of this attribute is 1, this attribute automatically unmounts VxFS checkpoints when the file system is unmounted. If the value of this attribute is 0, and checkpoints are mounted, then failover does not occur. Type and dimension: integer-scalar Default: 1
MountOpt	Options for the `mount` command. Refer to the `mount` manual page for more information. Type and dimension: string-scalar Example: "rw"
SecondLevelMonitor	This attribute is only applicable to NFS client mounts. If the value of this attribute is 1, this attribute enables detailed monitoring of an NFS mounted file system. The agent executes the `df -k` command for the NFS mounted file system to detect network outage. Type and dimension: boolean-scalar Default: 0
SecondLevelTimeout	This attribute is only applicable to NFS client mounts. This attribute is the timeout (in seconds) for the SecondLevelMonitor attribute. This attribute is only functional when the value of the SecondLevelMonitor attribute is 1. The actual timeout value can be much smaller. This setting depends on how much time remains before it exceeds the MonitorTimeout interval. Type and dimension: integer-scalar Default: 30

Table 2-12 Optional attributes

Optional attribute	Description
SnapUmount	If the value of this attribute is 1, this attribute automatically unmounts VxFS snapshots when the file system is unmounted. If the value of this attribute is 0 and snapshots are mounted, then failover does not occur. Type and dimension: integer-scalar Default: 0

Resource type definition

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

Mount agent notes

The Mount agent has the following notes:

- "High availability fire drill" on page 63
- "VxFS file system lock" on page 64

High availability fire drill

The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node; discrepancies that

might prevent a service group from going online on a specific node. For Mount resources, the high availability drill performs the following, it:

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

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

VxFS file system lock

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

```
# mount
```

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

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

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

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

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

Sample configurations

Configuration

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

ChapterX 3

Network agents

This chapter contains the following:

About the network agents

Use network agents to provide high availability for networking resources.

Agent comparisons

IP and NIC agents

The IP and NIC agents:

- Monitor a single NIC

IPMultiNIC and MultiNICA agents

The IPMultiNIC and MultiNICA agents:

- Monitor single or multiple NICs

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

IPMultiNICB and MultiNICB agents

The IPMultiNICB and MultiNICB agents:

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

802.1Q trunking

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

To use 802.1Q trunking, create 802.1Q trunked interfaces over a physical interface using the System Management Homepage (SMH). The physical interface is connected to a 802.1Q trunked port on the switch.

The NIC, and MultiNICA agents can monitor these trunked interfaces. The IP and IPMultiNIC agents monitor the virtual IP addresses that are configured on these interfaces.

For example, create a 802.1Q interface called lan9000 over a physical interface called lan0. Do not configure an IP address on lan0. You connect lan0 to a trunked port on the switch. The NIC and IP agents can then monitor lan9000 and the virtual IP address configured on lan9000. You must make sure that the IP addresses that are assigned to the interfaces of a particular VLAN are in the same subnet.

IP agent

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

The interface must be enabled with a physical (or administrative) base IP address before you can assign it a virtual IP address.

For the IP and NIC agents, VCS supports Auto-port Aggregation (APA).

High availability fire drill

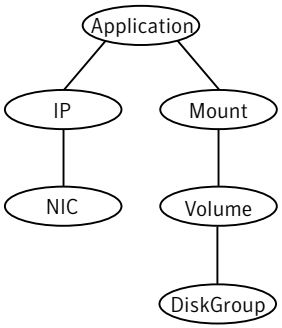
The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node. These discrepancies might prevent a service group from going online on a specific node. For IP resources, the high availability fire drill checks for the existence of a route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

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

Dependencies

IP resources depend on NIC resources.

Figure 3-1 Sample service group that includes an IP resource



Agent functions

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

State definitions

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

Attributes

Table 3-1 Required attributes

Required attributes	Description
Address	A virtual IP address, which is different from the base IP address, and which is associated with the interface. Note that the address you specify must not be the same as the configured physical IP address, but should be on the same network. Type and dimension: string-scalar Example: "192.203.47.61"

Table 3-1 Required attributes

Required attributes	Description
Device	The name of the NIC device that is associated with the IP address. Contains the device name without an alias. Type and dimension: string-scalar Example: "lan0"

Table 3-2 Optional attributes

Optional attributes	Description
ArpDelay	The number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about this IP address. Type and dimension: integer-scalar Default: 1
IfconfigTwice	Causes an IP address to be configured twice using an `ifconfig up-down-up` command sequence. Increases the probability of gratuitous ARP requests (generated by `ifconfig up`) to reach clients. Type and dimension: integer-scalar
NetMask	The netmask that is associated with the IP address of the resource. Specify the value of the netmask in decimal (base 10) or hexadecimal (base 16). **Note:** Symantec 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"

Table 3-2 Optional attributes

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

Resource type definition

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

Sample configurations

Configuration 1

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

NetMask in decimal (base 10)

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

NetMask in hexadecimal (base 16)

```
IP ipres (
    Device = lan0
    Address = "192.203.47.61"
    NetMask = "0xfffff800"
    )
```

NIC agent

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

For the NIC and IP agents, VCS supports Auto-port Aggregation (APA).

High availability fire drill

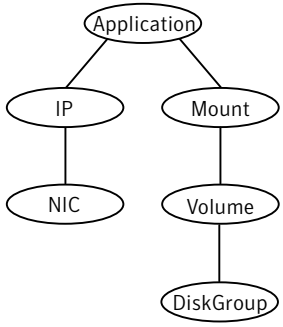
The high availability fire drill detects discrepancies between the VCS configuration and the underlying infrastructure on a node. These discrepancies might prevent a service group from going online on a specific node. For NIC resources, the high availability fire drill checks for the existence of the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

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

Dependencies

No child dependencies exist for this resource.

Figure 3-2 Sample service group that includes a NIC resource



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

Before you use this agent:

- Verify that the NIC has the correct administrative IP address and subnet mask.

- Verify that the NIC does not have built-in failover support. If it does, disable it.

Auto Port Aggregation (APA) support

HP APA aggregates multiple network interfaces so that they appear as a single interface. For example you can combine lan0 and lan1 and call the combined interface lan9000. You then use the NIC agent to monitor the lan9000 interface. You use the IP agent to configure and monitor an IP address on the lan9000 interface. Note that you use the lan9000 interface configured through APA for the Device attribute.

The IP and NIC agents support APA use with VCS. APA is responsible for providing local adapter swapping, which is outside of VCS control.

Agent functions

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

State definitions

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

Attributes

Table 3-3 Required attributes

Required attribute	Description
Device	Name of the NIC that you want to monitor. Type and dimension: string-scalar Example: "lan0"

Table 3-4 Optional attributes

Optional attribute	Description
NetworkHosts	List of hosts on the network that are pinged to determine if the network connection is alive. Enter the IP address of the host, instead of the host name, to prevent the monitor from timing out. DNS causes the ping to hang. If more than one network host is listed, the monitor returns ONLINE if at least one of the hosts is alive. Type and dimension: string-vector Example: "166.96.15.22" , "166.97.1.2"
NetworkType	Type of network. VCS currently only supports Ethernet. Type and dimension: string-scalar Default: "ether"

Table 3-4 Optional attributes

Optional attribute	Description
PingOptimize	Allows or disallows broadcast pings to control the network traffic that the NIC agent generates. Use the PingOptimize attribute when you have not defined a value for the NetworkHosts attribute. A value of 1 optimizes broadcast pings—it disallows the NIC agent from sending broadcast ping requests. A value of 0 tells the agent to perform a broadcast ping during each monitor cycle and detects the inactive interface. Type and dimension: integer-scalar Default: 1

Resource type definition

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

Sample configurations

Network Hosts

```
NIC groupx_lan0 (
    Device = lan0
    NetworkHosts = { "166.93.2.1", "166.99.1.2" }
    )
```

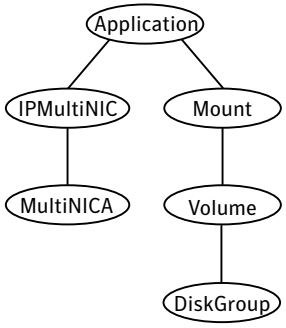
IPMultiNIC agent

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

Dependencies

IPMultiNIC resources depend on MultiNICA resources.

Figure 3-3 Sample service group that includes an IPMultiNIC resource



Agent functions

Online	Configures a virtual IP address on one interface of the MultiNICA resource.
Offline	Removes the virtual IP address from one interface of the MultiNICA resource.
Monitor	Checks if the virtual IP address is configured on one interface of the MultiNICA resource.
Clean	Removes 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 agent can not determine the state of the resource. This state may be due to an incorrect configuration.

Attributes

Table 3-5 Required attributes

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

Table 3-6 Optional attributes

Optional attribute	Description
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

Table 3-6 Optional attributes

Optional attribute	Description
NetMask	The netmask that is associated with the IP address of the resource. Specify the value of the netmask in decimal (base 10) or hexadecimal (base 16). **Note:** Symantec 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"

Resource type definition

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

MultiNICA agent

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

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

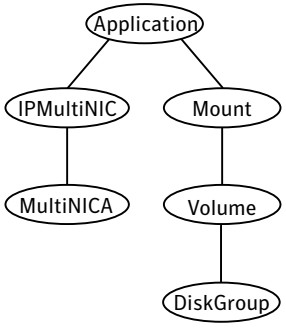
For important information on this agent, refer to:

See "MultiNICA notes" on page 85.

Dependencies

No dependencies exist for the MultiNICA resource.

Figure 3-4 Sample service group that includes a MultiNICA resource



Agent function

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

State definitions

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

Attributes

Table 3-7 Required attributes

Required attribute	Description
Device	List of interfaces and their base IP addresses. Type and dimension: string-association Example: { lan0 = "192.205.8.42", lan3 = "192.205.8.42" }

Table 3-8 Optional attributes

Optional attribute	Description
ArpDelay	Number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about the base IP address. Type and dimension: integer-scalar Default: 1
HandshakeInterval	Computes the maximum number of tries that 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 it marks a NIC as FAULTED. Increased values result in longer failover times, whether between the NICs or from system to system in the case of FAULTED NICs. Type and dimension: integer-scalar Default: 20 This value is the equivalent to two tries.

Table 3-8 Optional attributes

Optional attribute	Description
IfconfigTwice	Causes an IP address to be configured twice, using an `ifconfig up-down-up` sequence. Increases the probability of gratuitous ARP requests (caused by `ifconfig up`) to reach clients. Type and dimension: integer-scalar
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 host name, 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:** Symantec 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	A value of 1 indicates that the agent not perform broadcast pings. 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

Table 3-8 Optional attributes

Optional attribute	Description
RetestInterval	Number of seconds to sleep between re-tests of a newly configured interface. A lower value results in faster local (interface-to-interface) failover. Type and dimension: integer-scalar Default: 5
RouteOptions	String to add a route when configuring an interface. Use only when configuring the local host as the default gateway. The string contains destination gateway metric. No routes are added if this string is set to NULL. Type and dimension: string-scalar Example: "default 192.98.16.103 0"

Resource type definition

```
type MultiNICA (
    static str ArgList[] = { Device, NetMask, ArpDelay,
    RetestInterval, Options, RouteOptions, PingOptimize,
    MonitorOnly, IfconfigTwice, HandshakeInterval, NetworkHosts }
    static int 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
    int HandshakeInterval = 20
    str NetworkHosts[]
)
```

MultiNICA notes

General notes

- In a MultiNICA resource configuration, the link-local type of IPv6 addresses are not supported as the base address for device. The resource may enter a FAULTED state if the configuration contains a link-local address as a base address.
- If all NICs configured in the Device attribute are down, the MultiNICA agent faults the resource after a two-three minute interval. This delay occurs because the MultiNICA agent tests the failed NIC several times before it marks the resource OFFLINE. Failover logs record a detailed description of the events.
- The MultiNICA agent supports only one active NIC on one IP subnet; the agent does not work with multiple active NICs on the same subnet.
  - On HP-UX, for example, you have one active NIC, lan0 (10.128.2.5). 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. The redirect makes the MultiNICA monitor return an online status. Note that using ping -i does not enable the use of multiple active NICs.
- Before you start VCS, configure the primary NIC with the correct broadcast address and netmask.

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 configurations

MultiNICA and IPMultiNIC

In the following example, two systems, sysa and sysb, each have a pair of network interfaces, `lan0` and `lan3`. In this example, the two interfaces, `lan0` and `lan3`, have the same base, or physical, IP address. Note the lines beginning Device@sysa and Device@sysb; the use of different physical addresses shows how to localize an attribute for a particular host.

The MultiNICA resource fails over only the physical IP address to the backup NIC during a failure. The IPMultiNIC agent configures the logical IP addresses. The resources ip1 and ip2, shown in the following example, have the Address attribute 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. The Proxy resource points to the MultiNICA resource in the first group. The Proxy resource prevents redundant monitoring of the NICs on the same system. The IPMultiNIC resource is always made dependent on the MultiNICA resource. See the IPMultiNIC agent for more information.

```
group grp1 (
    SystemList = { sysa, sysb }
    AutoStartList = { sysa }
    )
    MultiNICA mnic (
      Device@sysa = { 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
```

About the IPMultiNICB and MultiNICB agents

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

Checklist to ensure the proper operation of MultiNICB

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

- Each interface must have a unique MAC address.
- A MultiNICB resource controls all the interfaces on one IP subnet.
- At boot time, you must configure and connect all the interfaces that are under the MultiNICB resource and give them test IP addresses.
- All test IP addresses for the MultiNICB resource must belong to the same subnet as the virtual IP address.
- Reserve the base IP addresses, which the agent uses to test the link status, for use by the agent. These IP addresses do not get failed over.
- If you specify the NetworkHosts attribute, then that host must be on the same subnet as the other IP addresses for the MultiNICB resource.

IPMultiNICB agent

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

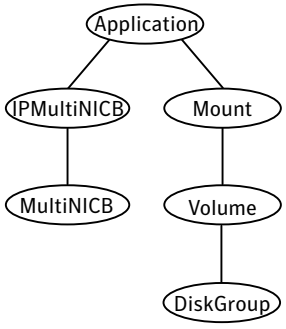
If multiple service groups have IPMultiNICB resources associated with the same MultiNICB resource, only one group should have a MultiNICB resource. The other groups should have a proxy resource pointing to the MultiNICB resource.

For the MultiNICB and IPMultiNICB agents, VCS supports Auto-port Aggregation (APA).

Dependencies

IPMultiNICB resources depend on MultiNICB resources.

Figure 3-5 Sample service group that includes an IPMultiNICB resource



Requirements for IPMultiNICB

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

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

Agent functions

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

State definitions

ONLINE	Indicates that an IP address on one of the working network interfaces of the resource is up. The IP address is specified in the Address attribute. The resource is specified in the BaseResName attribute.
OFFLINE	Indicates that an IP address on one of the working network interfaces of the resource is up. The IP address is specified in the Address attribute. The resource is specified in the BaseResName attribute.
UNKNOWN	Indicates that the agent cannot determine the status of the virtual IP address that is specified in the Address attribute.

Attributes

Table 3-9 Required attributes

Required attribute	Description
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"

Table 3-10 Optional attributes

Optional attribute	Description
DeviceChoice	Indicates the preferred NIC where you want to bring the logical IP address online. Specify the device name or NIC alias as determined in the Device attribute of the MultiNICB resource. Type and dimension: string-scalar Default: 0 Examples: DeviceChoice = "lan0" 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:** Symantec recommends that you specify a netmask for each virtual interface. Type and dimension: string-scalar

Table 3-10 Optional attributes

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

Resource type definition

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

Manually migrating a logical IP address

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

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

```
# haipswitch -s MultiNICB_resname
```

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

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

Sample configurations

Other sample configurations for IPMultiNICB and MultiNICB

Refer to the sample configurations in the MultiNICB agent.

MultiNICB agent

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

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

For the MultiNICB and IPMultiNICB agents, VCS supports Auto-port Aggregation (APA).

About the MultiNICB agent

The agent sends packets to other hosts on the network to monitor the interfaces that it controls. It then checks the link status of the interfaces.

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

Each NIC must have its own unique and exclusive base IP address, which the 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 works:

- Interface status: The interface status as reported by the driver of the interface (assuming that 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 the agent sends and receives 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.

Auto Port Aggregation (APA) support

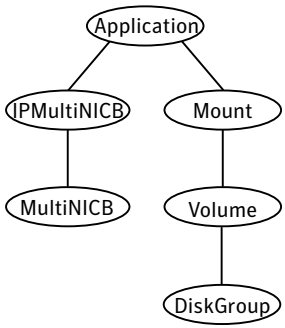
HP APA aggregates multiple network interfaces so that they appear as a single interface. For example you can combine lan0 and lan1 and call the combined interface lan9000. You then use the NIC agent to monitor the lan9000 interface. You use the IP agent to configure and monitor an IP address on the lan9000 interface. Note that you use the lan9000 interface configured through APA for the Device attribute.

The IP and NIC agents support APA use with VCS. APA is responsible for providing local adapter swapping, which is outside of VCS control.

Dependencies

No dependencies exist for the MultiNICB resource.

Figure 3-6 Sample service group that includes a MultiNICB resource



Agent functions

Open	Allocates an internal structure to store information about the resource.
Close	Frees the internal structure that is used to store information about the resource.
Monitor	Checks the status of each physical interface. Writes the status information to the export information file for IPMultiNICB resources to read it.

State definitions

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

Attributes

Table 3-11 Required attribute

Required attribute	Description
Device	List of NICs that you want under MultiNICB control, and the aliases of those NICs. The IPMultiNICB agent uses the NIC aliases to configure IP addresses. The IPMultiNICB agent uses these interface aliases to determine the order of the interface on which to bring the IP addresses online. Type and dimension: string-association 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.

Table 3-12 Optional attributes

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

Table 3-12 Optional attributes

Optional attribute	Description
Failback	If the value of the attract is 1, the virtual IP addresses are failed back to the original physical interface whenever possible. A value of 0 disables this behavior. Type and dimension: integer-scalar Default: 0
IgnoreLinkStatus	If the value of the attribute is 1, the agent ignores the driver-reported interface status while testing the interfaces. If the value of the attribute is 0, the agent reports the interface status as down if the driver-reported interface status indicates the down state. Using interface status for link testing may considerably speed up failovers. Type and dimension: integer-scalar Default: 1
LinkTestRatio	This attribute is the ratio of total monitor cycles to monitor cycles in which the agent tests the interfaces by sending packets. At all other times, the agent tests the link by checking the "link-status" as reported by the device driver. Checking the "link-status" is a faster way to check the interfaces, but only detects cable disconnection failures. If the value of the attribute is 1, packets are sent during every monitor cycle. If the value of the attribute is 0, packets are never sent during a monitor cycle. Type and dimension: integer-scalar Default: 1 Example: "3" In this example, if the monitor agent function invokes in a numbered pattern such as 1, 2, 3, 4, 5, 6, ..., the actual packet send test is done at 3, 6, ... monitor agent functions. For LinkTestRatio=4, the packet send test is done at 4, 8, ... monitor agent functions.

Table 3-12 Optional attributes

Optional attribute	Description
NetworkHosts	List of host IP addresses on the IP subnet that are pinged to determine if the interfaces work. NetworkHosts only accepts IP addresses to avoid DNS lookup delays. The IP addresses must be directly present on the IP subnet of interfaces (the hosts must respond to ARP requests). If IP addresses are not provided, the hosts are automatically determined by sending a broadcast ping (unless the NoBroadcast attribute is set to 1). The first host to reply serves as the ping destination. Type and dimension: string-vector Example: "192.1.0.1"
NetworkTimeout	Timeout for ARP and ICMP packets in milliseconds. MultiNICB waits for response to ICMP and ARP packets only during this time period. Assign NetworkTimeout a value in the order of tens of milliseconds (given the ICMP and ARP destinations are required to be on the local network). Increasing this value increases the time for failover. Type and dimension: integer-scalar Default: 100
NoBroadcast	If the value of the attribute is 1, NoBroadcast prevents MultiNICB from sending broadcast ICMP packets. (Note: MultiNICB can still send ARP requests.) If NetworkHosts are not specified and NoBroadcast is set to 1, the MultiNICB agent cannot function properly. **Note:** Symantec does not recommend setting the value of NoBroadcast to 1. Type and dimension: integer-scalar Default: 0

Table 3-12 Optional attributes

Optional attribute	Description
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 test helps to avoid oscillations in the status of the interface. Type and dimension: integer-scalar Default: 3

Resource type definition

```
type MultiNICB (
    static int MonitorInterval = 10
    static int OfflineMonitorInterval = 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
)
```

Trigger script

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

- MultiNICB resource name
- The device whose status changed, for example:
  - lan0
- The device's previous status (0 for down, 1 for up)
- The device's current status and monitor heartbeat

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

The sample script does the following:

- If interface changes status, it prints a message to the console, for example: MultiNICB: Device lan0 status changed from down to up.

IPMultiNICB and MultiNICB 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
```

DNS agent

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

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

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

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

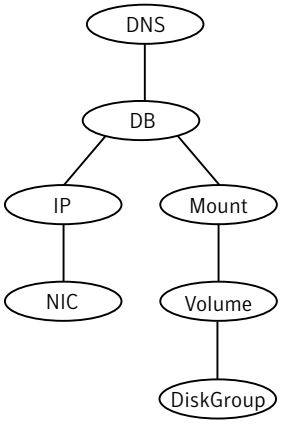
For important information about this agent, refer to:

"DNS agent notes" on page 110

Dependencies

No dependencies exist for the DNS resource.

Figure 3-7 Sample service group that includes a DNS resource



Agent functions

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

State definitions

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

Attributes

Table 3-13 Required attributes

Required attribute	Description
Domain	A string representing the DNS zone that the agent administers. The domain name can only contain alphanumeric symbols and the dash. Type and dimension: string-scalar Examples: ■ Forward mapping "demo.example.com" ■ IPv4 reverse mapping "2.168.192.in-addr.arpa"
■ Hostname and Alias or ■ ResRecord	You must use either the ResRecord attribute only or the HostName and Alias attributes. Do not use all three attributes together.
Alias	A string representing the alias to the canonical name. Type and dimension: string-scalar Example: "www" Where www is the alias to the canonical name location1.example.com. See "Sample Web server configuration" on page 111.
Hostname	A string that represents the canonical name of a system. Type and dimension: string-scalar Example: "location1.example.com"

Table 3-13 Required attributes

Required attribute	Description
ResRecord	You can use the ResRecord attribute alone, or you can use the Hostname and Alias attributes. ResRecord is an association of DNS resource record values. Each ResRecord attribute consists of two values: *DNS record key* = *DNS record data*. Note that the record key must be a unique value. If the resource record list contains any invalid value as a part of the record key or a record value of the ResRecord attribute, the resource enters an UNKNOWN state. Type and dimension: association-scalar Examples: ■ For forward mapping, where the zone is demo.example.com: - sles901 = "192.168.2.191" - ww2 = sles901 - sles9ip6 = "2007::1:2:3:abc" ■ A multi-home DNS record, typically for one host with two network interfaces, different address, but the same DNS name. This results in two-A records, or a single A record with continuation lines. sle902 = "192.168.2.102 10.87.13.22" A multi-home AAAA DNS record can be configured as below: sle902 = "1234::5678 1234::AABB:CCDD" ■ For reverse IPv4 address mapping, where the zone is 2.168.192.in-addr.arpa: 191 = "sles901.demo.example.com" ■ For reverse IPv6 address mapping, where the zone is 3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.7.0.0.2.ip6.arpa: cba = "sles9ip6.demo.example.com" Use only partial host names. If you use a fully qualified domain name, append a period "." at the end of the name. For CNAME records, use: ■ ResRecord = { www = mydesktop } or ■ ResRecord = { www = "mydesktop.marketing.example.com." } Where the Domain attribute is "marketing.example.com"

Table 3-14 Required attributes

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

Table 3-15 Optional attributes

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

Table 3-15 Optional attributes

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

Resource type definition

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

DNS agent notes

The DNS agent has the following notes:

High availability fire drill

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

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

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

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

Monitor scenarios

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

Table 3-16 Monitor scenarios for the Online lock file

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

Sample Web server configuration

Take the former Veritas corporate web server as an example. A browser requests the URL http://www.example.com that maps to the canonical name location1.example.com. The browser retrieves the IP address for the web server by querying a domain name server. If the web server fails over from location one to location two (location2.example.com), the domain name servers need a new canonical name mapping for www.example.com. The www.example.com alias is now updated to point to the canonical name of the standby system in location two.

Secure DNS update for BIND 9

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

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

Setting up secure updates using TSIG keys for BIND 9

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

To use secure updates using TSIG keys

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

   ```
   # dnssec-keygen -a HMAC-MD5 -b 2 -n HOST example.com.
       example.com.+157+00000
   ```

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

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

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

   ```
   +Cdjlkef9ZTSeixERZ433Q==
   ```

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

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

   Where **+Cdjlkef9ZTSeixERZ433Q==** is the key.

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

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

6. Save and restart the named process.

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

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

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

Chapter 4

File share agents

This chapter contains the following:

About the file service agents

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

NFS agent

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

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

Dependencies

For more information regarding NFS resource dependencies, refer to the *Veritas Cluster Server User's Guide*.

Figure 4-1 Sample service group that includes an NFS resource

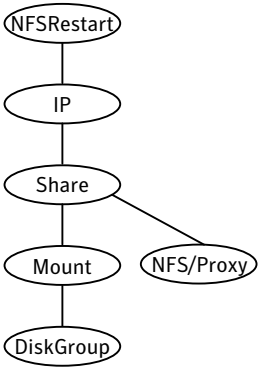
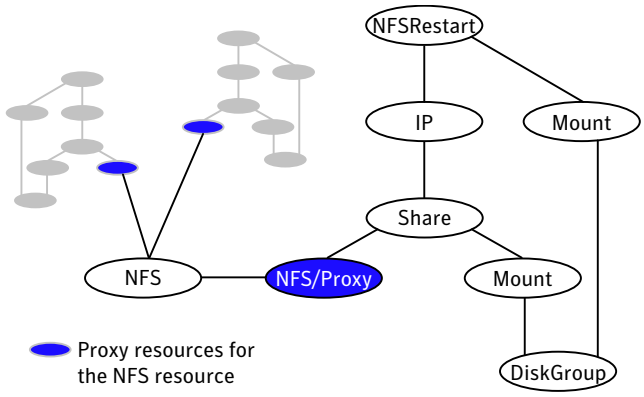


Figure 4-2 Using Proxy resources for multiple service groups that point to a single NFS resource to reduce redundant monitoring of the NFS daemons.



Agent functions

Online	Checks if nfsd and rpc.mountd daemons are running. If they are not running, the agent starts the daemons.
Monitor	Monitors versions 2, 3, and 4 of the nfsd daemons, and versions 1 and 3 of the rpc.mountd daemon. Monitors TCP and UDP versions of the daemons 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	Terminates and restarts the nfsd and rpc.mountd daemons.

State definitions

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

Attributes

Table 4-1 Optional attributes

Optional attribute	Description
LockFileTimeout	If the group goes offline, the agents waits for the specified time before restarting nfsd and rpc.mountd daemons. Type and dimension: integer-scalar Default: 180 (lower limit: 90 seconds, upper limit: 300 seconds)
Nservers	Specifies the number of concurrent NFS requests the server can handle. Type and dimension: integer-scalar Default: 4 Example: 24
Protocol	Selects the transport protocol that the NFS server supports. Allowed values are: tcp, udp or all. If you define Protocol to equal all, the NFS server supports both protocols. Type and dimension: string-scalar Default: all Example: "tcp”
Version	This attribute defines the maximum version of NFS that the agent supports. Type and dimension: integer-scalar Example: "3"

Resource type definition

```
type NFS (
    static int RestartLimit = 1
    static str ArgList[] = { Nservers, Protocol, LockFileTimeout,
    Version}
    static str Operations = OnOnly
    int Nservers = 4
    str Protocol = all
    int LockFileTimeout = 180
    int Version = 3
)
```

Sample configurations

Configuration

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

NFSRestart agent

The agent starts rpc.lockd and rpc.statd, which are responsible to send notification after reading the nfs lock directory. The agent copies some records in nfs lock directory, which is used by the re-started lockd or statd daemons for notification.

The smsyncd daemon copies lock information from /var/statmon/sm to shared storage. The agent's online function copies the locks from shared storage to /var/statmon/sm.

For important information about this agent, refer to:

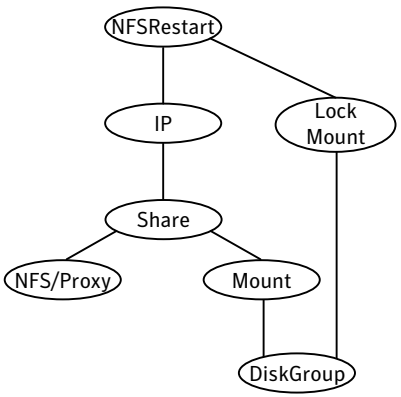
"NFSRestart agent notes" on page 120

Dependencies

For more information regarding NFSRestart resource dependencies, refer to the *Veritas Cluster Server User's Guide*.

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

Figure 4-3 Sample service group that includes an NFSRestart resource



Agent functions

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

State definitions

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

Attributes

Table 4-2 Required attributes

Required attribute	Description
LocksPathName	The path name of the directory to store the NFS lock information for all the shared filesystems. This attribute is mandatory when 'NFSLockFailOver = 1'. Type and dimension: string-scalar
NFSLockFailOver	A flag that specifies whether the user wants NFS Locks to be recovered after a failover Type and dimension: boolean-scalar Default: 0
NFSRes	Name of the NFS resource. Type and dimension: string-scalar

Resource type definition

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

NFSRestart agent notes

The NFSRestart agent has the following notes:

- "About high availability fire drill" on page 121
- "Providing a fully qualified host name" on page 121

About high availability fire drill

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

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

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

Providing a fully qualified host name

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

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

```
/etc/hosts
```

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

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

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

For example:

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

You have to make sure that the NFS client stores the same information for the NFS server as the client uses while mounting the file system. For example, if the NFS client mounts the file system using fully qualified domain names for the NFS server, then the NFS client directory: /var/lib/nfs/ directory should also have a fully qualified domain name after the acquisition of locks. Otherwise, you need to start and stop the NFS client twice using the /etc/init.d/nfs.client script to clear the lock cache of the NFS client.

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

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

Sample configurations

Basic NFSRestart configuration

```
include "types.cf"

cluster vcs_test (
        )

system sysA (
        )

system sysB (
        )

group NFSgrp1 (
        SystemList = { sysA = 0, sysB = 1 }
        AutoStartList = { sysA, sysB }
        )

        DiskGroup dg01 (
                DiskGroup = dg01
                StartVolumes = 0
                StopVolumes = 0
                )

        IP ip1 (
                Device = lan0
                Address = "11.123.175.11"
                NetMask = "255.255.248.0"
                )

        Mount Mount_dir1 (
                MountPoint = "/dir1"
                BlockDevice = "/dev/vx/dsk/dg01/vol01"
                FSType = vxfs
                MountOpt =
                "ioerror=mwdisable,largefiles,qio,delaylog"
                FsckOpt = "-n"
                )

        NFS nfs1 (
                Nservers = 8
                LockFileTimeout= 360
```

```
        )

NFSRestart nfsres1 (
        LocksPathName = "/dir1"
        NFSLockFailOver = 1
        NFSRes = nfs1
        )

NIC nic1 (
        Device = lan0
        NetworkHosts = {"11.123.170.107"}
        )

Share Share_dir1 (
        PathName = "/dir1"
        )

Volume dg01_vol01 (
        Volume = vol01
        DiskGroup = dg01
        )

ip1 requires nic1
ip1 requires Share_dir1
Mount_dir1 requires dg01_vol01
nfsres1 requires ip1
Share_dir1 requires Mount_dir1
Share_dir1 requires nfs1
dg01_vol01 requires dg01
```

Share agent

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

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

For important information on this agent, refer to:

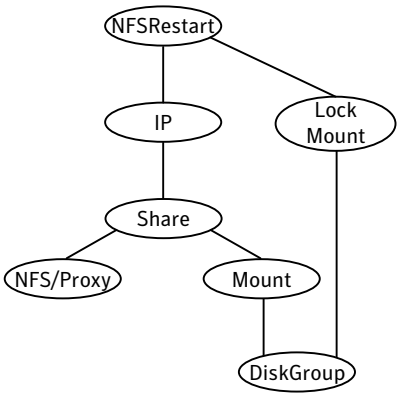
"Share agent notes" on page 127

Dependencies

For more information regarding Share resource dependencies, refer to the *Veritas Cluster Server User's Guide*.

Share resources depend on NFS. In an NFS service group, the IP family of resources depends on Share resources.

Figure 4-4 Sample service group that include a Share resource



Agent functions

Online	Shares an NFS file system.
Offline	Unshares an NFS file system.
Monitor	Reads the /etc/dfs/sharetab file and looks for an entry for the file system specified by PathName. If the entry exists, monitor returns ONLINE.

Action	direxists.vfd Checks if the path specified by the PathName attribute exists on the cluster node. If the path name is not specified, it checks if a corresponding mount point is available to ensure that the path is on shared storage.

State definitions

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

Attributes

Table 4-3 Required attributes

Required attribute	Description
PathName	Pathname of the file system to be shared. Type and dimension: string-scalar Example: "/share1x"
NFSRes	Name of the NFS resource. Type and dimension: string-scalar Example: "nfsres_01"

Table 4-4 Optional attributes

Optional attribute	Description
Options	Options for the `share` command. Type and dimension: string-scalar Examples: "-o ro" or "-o rw=*hostname*"

Resource type definition

```
type Share (
    static keylist SupportedActions = { "direxists.vfd" }
    static str ArgList[] = { PathName, Options, "NFSRes:State" }
    static int NumThreads = 1
    str NFSRes
    str PathName
    str Options
)
```

Share agent notes

The following section contains notes on the Share agent.

High availability fire drill

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

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

Sample configurations

Basic agent configuration

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

About the Samba agents

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

- Filespace
- Printer
- WINS
- Domain Master

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

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

The Samba agents

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

Before using the Samba agents

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

Supported versions

VCS supports most versions of Samba that are bundled with supported operating systems. For operating systems that do not come bundled with Samba, VCS supports most versions that are compatible with the operating system.

Configuring the Samba agents

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

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

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

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

SambaServer agent

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

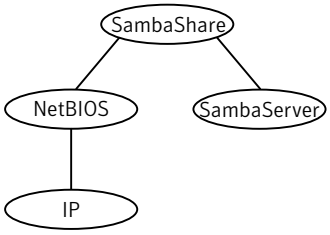
The smbd daemon provides Samba share services. The agent makes a copy of smbd for each client and verifies that Samba is running by reading the pid of this daemon. The agent can perform in-depth monitoring by establishing a socket connection to Samba at ports where the daemon is listening and sending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

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

Dependencies

No dependencies exist for the SambaServer resource. You can configure only one SambaServer resource on a node.

Figure 4-5 Sample service group that includes a SambaServer resource



Agent functions

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

State definitions

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

Attributes

Table 4-5 Required attributes

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

Resource type definitions

```
type SambaServer (
    static str ArgList[] = { ConfFile, LockDir, Ports,
    IndepthMonitorCyclePeriod, ResponseTimeout }
    str ConfFile
    str LockDir
    int Ports[] = { 139, 445 }
    int IndepthMonitorCyclePeriod = 5
    int ResponseTimeout = 10
)
```

Sample configurations

```
SambaServer Samba_SambaServer (
    ConfFile = "/etc/opt/samba/smb.conf"
    LockDir = "/var/opt/samba/locks"
    )
```

SambaShare agent

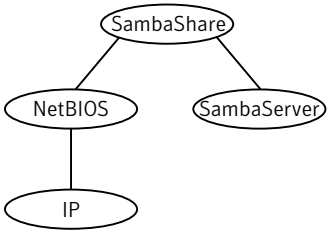
The SambaShare agent adds, removes, and monitors a share by modifying the specified Samba configuration file. You can use the agent to make a Samba Share highly available or to monitor it.

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

Dependencies

SambaShare resources depend on SambaServer, NetBios and Mount resources.

Figure 4-6 Sample service group for a SambaShare resource



Agent functions

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

State definitions

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

Attributes

Table 4-6 Required attributes

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

Resource type definition

You do not need to configure these attributes "SambaServerRes:ConfFile", "SambaServerRes:SambaTopDir" and "SambaServerRes:LockDir".

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

Sample configuration

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

NetBIOS agent

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

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

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

Before using this agent:

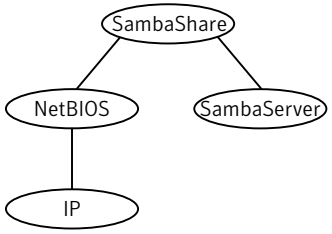
- Set the NetBIOS name.
- Set the NetBIOS interface.

Dependencies

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

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

Figure 4-7 Sample service group that includes a NetBIOS resource



Agent functions

Online	Updates the Samba configuration with the NetBIOS name, all NetBIOS aliases and network interfaces, WINS support, and domain master options specified in the NetBIOS resource. Starts the nmbd daemon.

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

State definitions

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

Attributes

Table 4-7 Required attributes

Required attribute	Description
NetBiosName	Name by which the Samba server is known in the network. Type and dimension: string-scalar

Table 4-8 Optional attributes

Optional attribute	Description
Interfaces	List of network interfaces on which Samba handles browsing. Type and dimension: string-vector Example: "172.29.9.24/16"
NetBiosAliases	List of additional names by which the Samba server is known in the network. Type and dimension: string-vector Example: "host1_samba, myname"
WinsSupport	If set to 1, this flag causes the agent to configure Samba as a WINS server. Type and dimension: integer-scalar Default: 0
DomainMaster	If the value of this attribute is 1, the agent sets Samba as Domain Master. Type and dimension: integer-scalar Default: 0

Resource type definition

You do not need to configure these attributes "SambaServerRes:ConfFile" and "SambaServerRes:LockDir".

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

Sample configuration

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

Chapter 5

Service and application agents

This chapter contains the following agents:

About the service and application agents

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

Apache Web server agent

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

This agent supports the Apache HTTP server 1.3, 2.0, and 2.2.

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

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

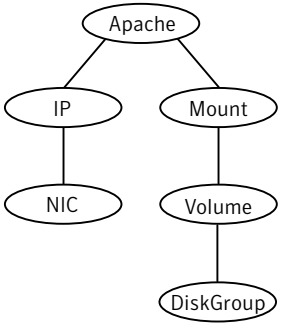
For more information regarding this agent:

See "Apache Web server notes" on page 149.

Dependencies

This type of resource depends on IP and Mount resources.

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



Agent functions

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

State definitions

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

Attributes

Table 5-1 Required attributes

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

Table 5-1 Required attributes

Required attribute	Description
PidFile	This attribute is required when you want to enable the detection of a graceful shutdown outside of VCS control. See "PidFile" on page 147.

Table 5-2 Optional attributes

Optional attribute	Description
DirectiveAfter	A list of directives that httpd processes after reading the configuration file. Type and dimension: string-association Example: DirectiveAfter{} = { KeepAlive=On }
DirectiveBefore	A list of directives that httpd processes before it reads the configuration file. Type and dimension: string-association Example: DirectiveBefore{} = { User=nobody, Group=nobody }
User	Account name the agent uses to execute the httpd program. If you do not specify this value, the agent executes httpd as the root user. Type and dimension: string-scalar Example: "apache1"
EnableSSL	Set to 1 (true) to have the online agent function add support for SSL by including the option `-DSSL` in the start command. For example: `/usr/sbin/httpd -f path_to_httpd.conf -k start -DSSL` Where path_to_httpd.conf file is the path to the httpd.conf file. Set to 0 (false) it excludes the `-DSSL` option from the command. Type and dimension: boolean-scalar Default: 0 Example: "1"

Table 5-2 Optional attributes

Optional attribute	Description
HostName	The virtual host name that is assigned to the Apache server instance. The host name is used in second-level monitoring for benchmarking the Apache HTTP server. **Note:** The HostName attribute is only required when the value of SecondLevelMonitor is 1 (true). Type and dimension: string-scalar Example: "web1.example.com"
Port	Port number where the Apache HTTP server instance listens. The port number is used in second-level monitoring for benchmarking the Apache HTTP server. Specify this attribute only if SecondLevelMonitor is set to 1 (true). Type and dimension: integer-scalar Default: 80 Example: "80"
EnvFile	Full path and file name of the file that is sourced before executing httpdDir/httpd. With Apache 2.0, the file *ServerRoot*/bin/envvars, which is supplied in most Apache 2.0 distributions, is commonly used to set the environment before executing httpd. Specifying this attribute is optional. If EnvFile is specified, the shell for user root must be Bourne, Korn, or C shell. Type and dimension: string-scalar Example: "/apache/server1/bin/envvars"
PidFile	The PidFile attribute sets the file to which the server records the process ID of the daemon. The value of PidFile attribute must be the absolute path where the Apache instance records the pid. This attribute is required when you want the agent to detect the graceful shutdown of the Web server. For the agent to detect the graceful shutdown of the Web server, the value of the IntentionalOffline resource type attribute must be 1 (true). Type and dimension: string-scalar Example: /var/run/httpd.pid

Table 5-2 Optional attributes

Optional attribute	Description
SharedObjDir	Full path of the directory in which the Apache HTTP shared object files are located. Specifying this attribute is optional. It is used when the HTTP Server is compiled using the SHARED_CORE rule. If you specify this attribute, the directory is passed to the `-R` option when executing the httpd program. Refer to the httpd man pages for more information about the `-R` option. Type and dimension: boolean-scalar Example: "/apache/server1/libexec"
SecondLevelTimeout	The number of seconds that the monitor agent function waits on the execution of second-level monitor. If the second-level monitor program does not return to calling the monitor agent function before the SecondLevelTimeout window expires, the monitor agent function no longer blocks on the program sub-process. It does, however, report that the resource is offline. The value should be high enough to allow the second level monitor enough time to complete. The value should be less than the value of the agent's MonitorTimeout. Type and dimension: integer-scalar Default: 30

Table 5-3 Resource type attribute

Required attribute	Description
IntentionalOffline	For information on how to use the IntentionalOffline resource type attribute, refer to the *Veritas Cluster Server User's Guide*.

Resource type definition

```
type Apache (
    static keylist SupportedActions = { "checkconffile.vfd" }
    static str ArgList[] = { ResLogLevel, State, IState, httpdDir,
    SharedObjDir, EnvFile, PidFile, HostName, Port, User,
    SecondLevelMonitor, SecondLevelTimeout, ConfigFile, EnableSSL,
    DirectiveAfter, DirectiveBefore }
    str ResLogLevel = INFO
```

```
    str httpdDir
    str SharedObjDir
    str EnvFile
    str PidFile
    str HostName
    int Port = 80
    str User
    boolean SecondLevelMonitor
    int SecondLevelTimeout = 30
    str ConfigFile
    boolean EnableSSL
    str DirectiveAfter{}
    str DirectiveBefore{}
    static int IntentionalOffline = 0
)
```

Apache Web server notes

The Apache Web server has the following notes:

- "Tasks to perform before you use the Apache Web server agent" on page 149
- "About detecting application failure" on page 150
- "About bringing an Apache Web server online outside of VCS control" on page 150
- "About high Availability fire drill" on page 151

Tasks to perform before you use the Apache Web server agent

Before you use this agent, perform the following tasks:

- Install the Apache server on shared or local disks.
- Ensure that you are able to start the Apache Web server outside of VCS control, with the specified parameters in the Apache configuration file (for example: /etc/apache/httpd.conf). For more information on how to start the server:
  See "About bringing an Apache Web server online outside of VCS control" on page 150.
- Specify the location of the error log file in the Apache configuration file for your convenience (for example: ErrorLog /var/apache/logs/error_log).
- Verify that the floating IP has the same subnet as the cluster systems.
- If you use a port other than the default 80, assign an exclusive port for the Apache server.

- Verify that the Apache server configuration files are identical on all cluster systems.
- Verify that the Apache server does not autostart on system startup.
- Verify that `Inetd` does not invoke the Apache server.
- Remove previous versions of this agent.
- The service group has disk and network resources to support the Apache server resource.
- Assign virtual host name and port to Apache Server.

About detecting application failure

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

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

About bringing an Apache Web server online outside of VCS control

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

To start an Apache Web server outside of VCS control

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

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

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

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

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

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

About high Availability fire drill

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

- ConfigFile
- httpdDir

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

Sample configurations

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

        Apache httpd_server (
                Critical = 0
                httpdDir = "/apache/bin"
                HostName = vcshp1
                Port = 8888
                User = root
                SecondLevelMonitor = 1
                ConfigFile = "/apache/conf/httpd.conf"
                )

        DiskGroup Apache_dg (
                Critical = 0
                DiskGroup = apc1
                )

        IP Apache_ip (
                Critical = 0
                Device = lan0
                Address = "11.123.99.168"
                NetMask = "255.255.254.0"
                )

        Mount Apache_mnt (
                Critical = 0
```

```
        MountPoint = "/apache"
        BlockDevice = "/dev/vx/dsk/apc1/apcvol1"
        FSType = vxfs
        FsckOpt = "-y"
        )

Apache_mnt requires Apache_dg
httpd_server requires Apache_mnt
httpd_server requires Apache_ip
```

Application agent

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

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

You can monitor the application in the following ways:

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

High availability fire drill

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

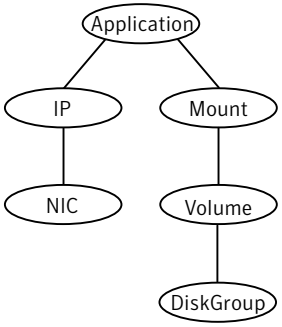
- The availability of the specified program
- Execution permissions for the specified program
- The existence of the specified user on the host
- The existence of the same binary on all nodes

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

Dependencies

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

Figure 5-2 Sample service group that includes an Application resource



Agent functions

Online — Runs the command or script that you specify in the value of the StartProgram attribute. Runs the command with the specified parameters in the context of the specified user.

Offline — Runs the command or script that you specify in the value of the StopProgram attribute. Runs the command with the specified parameters in the context of the specified user.

Monitor — If you specify the MonitorProgram attribute, the agent executes the user-defined MonitorProgram in the user-specified context. If you specify the PidFiles attribute, the routine verifies that the process ID that is found in each listed file is running. If you specify the MonitorProcesses attribute, the routine verifies that each listed process is running in the context you specify.

Use any combination among these attributes (MonitorProgram, PidFiles, or MonitorProcesses) to monitor the application.

If any of the processes that are specified in either PidFiles or MonitorProcesses is determined not to be running, the monitor returns OFFLINE. If the process terminates ungracefully, the monitor returns OFFLINE and failover occurs.

Clean — Terminates processes specified in PidFiles or MonitorProcesses. Ensures that only those processes (that are specified in the MonitorProcesses attribute) running with the user ID specified in the User attribute are killed. If the CleanProgram is defined, the agent executes the CleanProgram.

State definitions

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

Attributes

Table 5-4 Required attributes

Required attribute	Description
StartProgram	The executable, created locally on each node, which starts the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. **Note:** Do not use the opening and closing ({ }) brace symbols in this string. Type and dimension: string-scalar Example: "/usr/sbin/sample_app start"
StopProgram	The executable, created locally on each node, which stops the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. **Note:** Do not use the opening and closing ({ }) brace symbols in this string. 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 157.

Table 5-5 Optional attributes

Optional attribute	Description
CleanProgram	The executable, created locally on each node, which forcibly stops the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. 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 that the `ps -u user -o args | more` command displays 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 have spaces separating them. MonitorProgram can return the following VCSAgResState values: OFFLINE value is 100; online values range from 101 to 110 (depending on the confidence level); 110 equals confidence level of 100%. Any other value = UNKNOWN. **Note:** Do not use the opening and closing ({ }) brace symbols in this string. Type and dimension: string-scalar Example: "/usr/sbin/sample_app_monitor all"

Table 5-5 Optional attributes

Optional attribute	Description
PidFiles	A list of PID (process ID) files that contain the PID of the processes that you want monitored and cleaned. These are application generated files. Each PID file contains one monitored PID. Specify the complete path of each PID file in the list. The process ID can change when the process restarts. If the application takes time to update the PID file, the agent's Monitor function may return an incorrect result. If incorrect results occur, increase the ToleranceLimit in the resource definition. Type and dimension: string-vector Example: "/etc/sample/sample_app.pid"
User	The user ID for running StartProgram, StopProgram, MonitorProgram, and CleanProgram. The processes that are specified in the MonitorProcesses list must run in the context of the specified user. Monitor checks the processes to make sure they run in this context. Type and dimension: string-scalar Default: root

Resource type definition

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

Sample configurations

Sample Configuration 1

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

```
Application sendmail (
    User = root
    StartProgram = "/sbin/init.d/sendmail start"
    StopPragram = "/sbin/init.d/sendmail stop"
    PidFiles = {"/etc/mail/sendmail.pid"}
)
```

Sample Configuration 2

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

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

High availability fire drill

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

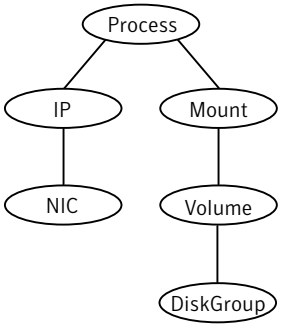
- The existence of the specified process
- Execution permissions for the specified process
- The existence of a binary executable for the specified process
- The existence of the same binary on all nodes

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

Dependencies

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

Figure 5-3 Sample service group for a Process resource



Agent functions

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

State definitions

ONLINE	Indicates that the specified process is running in the specified user context. The agent only reports the process as online if the value configured for PathName attribute exactly matches the process listing from the ps output.
OFFLINE	Indicates that the specified process is not running in the specified user context.
FAULTED	Indicates that the process has terminated unexpectedly.
UNKNOWN	Indicates that the agent can not determine the state of the process.

Attributes

Table 5-6 Required attribute

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

Table 5-7 Optional attributes

Optional attribute	Description
Arguments	Passes arguments to the process. If a script controls the process, the script is passed as an argument. Separate multiple arguments with a single space. A string cannot accommodate more than one space between arguments, nor allow for leading or trailing whitespace characters. This attribute must not exceed 80 characters. Type and dimension: string-scalar Example: "bd -q30m"
PidFile	File containing the process ID. Type and dimension: string-scalar Example: "/etc/mail/sendmail.pid"

Table 5-7 Optional attributes

Optional attribute	Description
Priority	Priority with which the process runs. 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"
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"

Resource type definition

```
type Process (
    static keylist SupportedActions = { "program.vfd", getcksum }
    static str ArgList[] = { PathName, Arguments, UserName,
    Priority, PidFile }
    str PathName
    str Arguments
    str UserName = root
    str Priority = 20
    str PidFile
)
```

Sample configurations

Configuration 1

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

Sample configuration 2

```
cluster vcs_test (
    UserNames = { admin = bIJbIDiFJeJJhRJdIG }
    Administrators = { admin }
    )

system sysA (
    )

system sysB (
    )
group ProcessGroup (
    SystemList = { sysa, sysb }
    AutoStartList = { sysa }
    )

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

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

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

ProcessOnOnly agent

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

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

Dependencies

No child dependencies exist for this resource.

Agent functions

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

State definitions

ONLINE	Indicates that the specified process is running.The agent only reports the process as ONLINE if the value configured for PathName attribute exactly matches the process listing from the ps output.
FAULTED	Indicates that the process has unexpectedly terminated.
UNKNOWN	Indicates that the agent can not determine the state of the process.

Attributes

Table 5-8 Required attributes

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

Table 5-9 Optional attributes

Optional attribute	Description
IgnoreArgs	A flag that indicates whether monitor ignores the argument list. ■ If the value is 0, it checks the process pathname and argument list. ■ If the value is 1, it only checks for the executable pathname and ignores the rest of the argument list. Type and dimension: boolean-scalar Default: 0
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 a total of 80 characters. Type and dimension: string-scalar Example: "-bd -q30m"

Resource type definition

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

HPVirtualMachine agent

The HPVirtualmachine agent brings online, takes offline, and monitors virtual machines (VMGuests) that are running on the physical host (VMHost).

Limitations

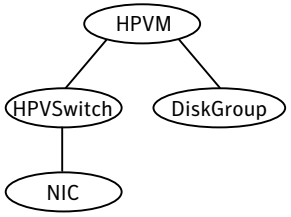
The HPVirtualMachine agent has the following lineations:

- The agent cannot detect if the operating system hangs. Even if the HPVirtualMachine reports the VMGuest state as ONLINE, it does not mean that the OS running within guest is functioning properly.

Dependencies

This resource depends on the DiskGroup agent and the HPVSwitch agent resources for its datastore and network.

Figure 5-4 Sample service group for a HPVirtualMachine resource, where HPVM represents the HPVirtualMachine resource



Agent functions

Online	Uses the `hpvmstart` command to start the virtual machine (VMGuest).
Offline	Attempts a graceful shut down of the virtual machine. Uses the `hpvmstop` command to stop the VMGuest.
Monitor	Uses the `hpvmstatus` command to detect the virtual machine's state. Returns the following status: ■ If the Virtual Machine is missing, it returns an UNKNOWN state. ■ If the Virtual Machine is running, then it returns an ONLINE state. ■ If the Virtual Machine is not running, it returns OFFLINE.
Clean	Forcefully shuts down the virtual machine. It uses `hpvmstop` command with `-h` argument.

State definitions

ONLINE	Indicates that the virtual machine (VMGuest) is up and has a heartbeat.
OFFLINE	Indicates that the virtual machine is turned off.
FAULTED	Indicates that the virtual machine has failed to start up using the online operation. This can occur due to an issue with the VMGuest configuration. It can also occur due to a sudden unexpected shutdown of the virtual machine.
UNKNOWN	Indicates the agent cannot determine the virtual machine's state. This state can occur if the virtual machine has not been created yet or the resource type definition of the HPVirtualMachine agent is not configured properly.

Attributes

Table 5-10 Required attributes

Required attribute	Description
VMName	The virtual machine (VMGuest) name that the agent monitors. This attribute is unique. Type-dimension: string-scalar Example: "vmg_01"
DelayAfterOnline	Defines the maximum time that the VMGuest can take to reach the EFI (Extensible Firmware Interface) shell. If the VMGuest leaves the EFI shell before the time provided by DelayAfterOnline, the online function exits at that time only. This attribute is added as to ensure that the boot time of VMGuest up to EFI shell remains flexible and can be modified as per the requirement. Type-dimension: integer-scalar Default: 30

Table 5-11 Resource type attribute

Required attribute	Description
IntentionalOffline	For information on how to use the IntentionalOffline resource type attribute, refer to the *Veritas Cluster Server User's Guide.*

Resource type definition

```
type HPVirtualMachine (
    static int IntentionalOffline = 1
    static str ArgList[] = { VMName, DelayAfterOnline, State, IState
    }
    str VMName
    int DelayAfterOnline = 30
)
```

Sample configurations

Basic HPVirtualMachine configuration

```
group group1 (
    SystemList = { system1 = 0, system = 2 }
    )
    HPVirtualMachine vm (
        VMName = vcsivm
        )
```

HPVirtualMachine configuration with an HPVSwitch resource

```
group group1 (
    SystemList = { system1 = 0, system = 2 }
    )
    HPVirtualMachine vm (
        VMName = vcsivm
        )

    HPVSwitch vswitch (
        VSwitchName = pub0
       )
    vm  requires vswitch
```

HPVSwitch agent

Use the HPVSwitch agent to manage and control the virtual switches that are associated with the network connection of the virtual machines (VMGuests). Virtual switches are virtual entities that resemble a normal switch to VMGuests.Virtual switches are mapped to the physical network interface card (NIC) on the physical machine (VMHost).

Note: If the virtual switch is started upon system reboot, you may observe a concurrency violation for the HPVSwitch resource that is configured as a part of a failover service group.

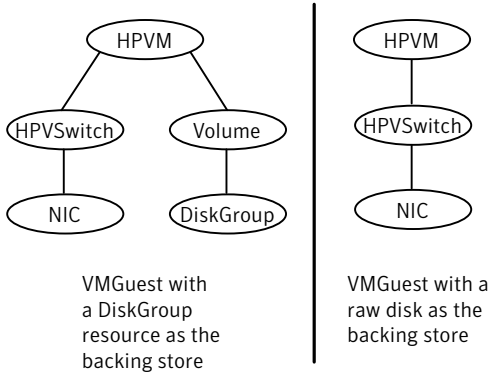
For agent limitations and requirements, see:

"HPVSwitch agent notes" on page 174

Dependencies

Virtual switches are mapped to the physical NIC on the physical machine (VMHost). The HPVSwitch resource depends on the NIC resource for its functionality. In the following diagrams, HPVM represents the HPVirtualMachine resource.

Figure 5-5 Sample service group for HPVSwitch resources



Agent functions

Online	Starts the virtual switch using the native `hpvmnet` commands. The online function requires the VSwitchName attribute to operate. When a switch is started, all the virtual NICs that are mapped to that virtual switch are activated.
Offline	Stops the virtual switch using the `hpvmnet` command. After the shutdown of VMGuest, no network is required. The native `hpvmnet` command halts the VSwitch.
Monitor	This function checks the status of the VSwitch and returns the following results: ■ If the virtual switch is running, it returns ONLINE. ■ If the virtual switch is not running or is halted, it returns OFFLINE. ■ If the virtual switch is not present on the system, it returns the state UNKNOWN.
Clean	Stops the virtual switch. It is similar to the offline operation.

State definitions

ONLINE	The VSwitch for the HPVSwitch agent is running properly.
OFFLINE	The VSwitch is halted. It is currently switched off.
UNKNOWN	The VSwitch is not configured properly.
FAULTED	The VSwitch has unexpectedly turned off or it failed to start.

Attributes

Table 5-12 Required attributes

Required attribute	Description
VSwitchName	Name of the VSwitch that the HPVSwitch agent monitors. This attribute is unique. Type-dimension: string-scalar Example: "vswitch_01"

HPVSwitch agent notes

The HPVSwitch agent has the following notes:

Agent limitation

This agent does not manage the virtual NIC on the VMGuest.

Requirements

For the HPVSwitch resource agents to work, the virtual switches must be present on the VMHost because the VMHost controls the virtual switches. The VMGuest has no information about the virtual switches.

Resource type definition

```
type HPVSwitch (
    static str ArgList[] = { VSwitchName }
    str VSwitchName
)
```

Sample configurations

The following sample configurations are for HP Integrity virtual machine service groups:

Creating an Integrity virtual machine service group

The following is a high-level overview of creating an Integrity virtual machine service group.

Configuring the failover service group

Configure a failover service group. Perform the following steps:

- Name the service group *IVM*.
- Make sure that the SystemList attribute contains all the cluster nodes where the VMGuest can failover.

Creating an HP virtual switch resource

Create and configure an HPVSwitch resource. Perform the following steps:

- Configure an HPVSwitch resource *vswitch* inside the IVM service group.
- Assign the virtual switch name that you want to monitor in the VSwitchName attribute.
- Create a NIC resource *nic* inside the IVM service group.
- Make sure that the Device attribute contains the virtual switch's backing physical NIC.
- Create a dependency between the parent HPVSwitch resource *vswitch* and its child NIC resource *nic*.

Creating the VMGuest resource

Create and configure the VMGuest resource. Perform the following steps:

- Configure a HPVirtualMachine resource *hpvm* inside the *IVM* service group.
- Add the name of the VMGuest that you want to monitor in the VMName attribute.
- Create a dependency between the parent HPVirtualMachine resource *hpvm* and its child resource *vswitch*.

Creating backing storage resources for the VMGuest resource

Create and configure the backing storage resources. Perform the following steps:

- If you intend to use a disk group as the backing store for the VMGuest, configure a DiskGroup resource *dg* with the disk group's name in the DiskGroup attribute. If you are use LVMs, raw disks, or CVM, use the corresponding storage resources.
- If the VMGuest is installed on a VxVM volume, create a Volume resource *vol*. Use the volume name for the backing storage for the VMGuest. Refer to the `hpvm` commands to learn more about backing storage for VMGuests.
- Create a dependency between the parent HPVirtualMachine resource *vm* and the child Volume resource *vol*.
- Ensure that all resources are enabled before bringing them online.

Service group with disk group as the backing store

```
include "types.cf"
cluster ivmclus (
    )
system sysA (
```

```
    system sysB (
        )
    group IVM (
        SystemList = { sysA = 0, sysB = 1 }
        )
        DiskGroup dg (
            DiskGroup = dg1
            )
        HPVSwitch pub0 (
            VswitchName = pub0
            )
        HPVirtualMachine hpvm (
            VMName = vcsivm1
            )
        NIC nic (
            Device = lan0
            )
        Volume vol (
            Volume = vol1
            DiskGroup = dg1
        )
        pub0 requires nic
        vm requires pub0
        vm requires vol
        vol requires dg

    // resource dependency tree
    //
    //    group IVM_OS
    //    {
    //    HPVirtualMachine hpvm
    //        {
    //        Volume vol
    //            {
    //            DiskGroup dg
    //            }
    //        HPVSwitch pub0
    //            {
    //            NIC nic
    //            }
    //        }
    //    }
```

Service group with raw disk as the backing store

```
include "types.cf"
cluster Test (
    )
system SysA (
    )
system SysB (
    )
```

```
group g1 (
    SystemList = { SysA = 0, SysB = 1 }
    )
    HPVSwitch vswitch (
        VSwitchName = public0
        )
    HPVirtualMachine hpvm (
        VMName = vmsharedhp
        )
hpvm requires vswitch
```

Service group with an online VM guest migration feature

```
include "types.cf"
cluster ivmclus (
    )
system sysA (
    )
system sysB (
    )
group IVM (
    SystemList = { sysA = 0, sysB = 1 }
    )
    HPVirtualMachine hpvm (
        VMName = vcsivm1
    )
```

Chapter 6

Infrastructure and support agents

This chapter contains the following agents:

About the infrastructure and support agents

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

NotifierMngr agent

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

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

Dependency

The NotifierMngr resource can depend on the NIC resource.

Agent functions

Online	Starts the notifier process with its required arguments.
Offline	VCS sends a `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.

Attributes

Table 6-1 Required attributes

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

Table 6-2 Optional attributes

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

Table 6-2 Optional attributes

Optional attribute	Description
MessagesQueue	Size of the VCS engine's message queue. Minimum value is 30. Type and dimension: integer-scalar Default: 30
NotifierListeningPort	Any valid, unused TCP/IP port number. Type and dimension: integer-scalar Default: 14144
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"
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

Table 6-2 Optional attributes

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

Table 6-2 Optional attributes

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

Resource type definition

```
type NotifierMngr (
    static int RestartLimit = 3
    static str ArgList[] = { EngineListeningPort, MessagesQueue,
    NotifierListeningPort, SnmpdTrapPort, SnmpCommunity,
    SnmpConsoles, 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{}
)
```

Sample configuration

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

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

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

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

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

Configuration

```
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
//              }
//      }
```

VRTSWebApp agent

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

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

Agent functions

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

State definitions

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

Attributes

Table 6-3 Required attributes

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

Resource type definition

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

Sample configuration

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

Proxy agent

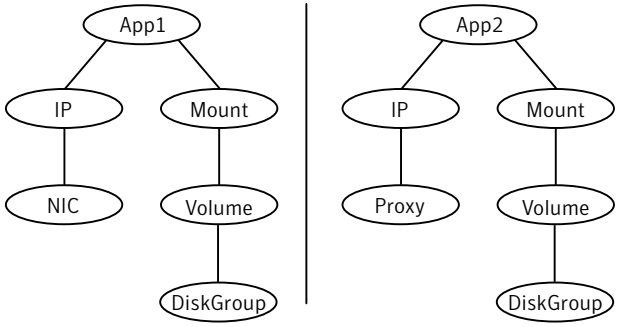
The Proxy agent mirrors the state of another resource on a local or remote system. It provides a means to specify and modify one resource and have its state reflected by its proxies. You can use the agent when you need to replicate the status of a resource.

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

Dependencies

No dependencies exist for the Proxy resource.

Figure 6-1 Sample service group that includes a Proxy resource



Agent functions

Monitor Determines status based on the target resource status.

Attributes

Table 6-4 Required attribute

Required attribute	Description
TargetResName	Name of the target resource that the Proxy resource mirrors. The target resource must be in a different resource group than the Proxy resource. Type and dimension: string-scalar Example: "tmp_VRTSvcs_file1"

Table 6-5 Optional attribute

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

Resource type definition

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

Sample configurations

Configuration 1

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

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

Configuration 2

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

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

Configuration

The proxy resource mirrors the state of the resource mnic on the local system; note that target resource is in grp1, and the proxy is 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
```

Phantom agent

The Phantom agent enables VCS to determine the state of parallel service groups that do not include OnOff resources.

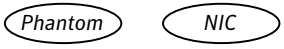
Do not use the Phantom resource in failover service groups.

Note: Do not attempt manual online or offline operations on the Phantom resource at the resource level. Do not use `hares` commands on the Phantom resource at the resource level. Unpredictable behavior results when you try a manual online or offline procedure or an `hares` command on a Phantom resource. You can perform commands on the service group that contains the Phantom resource.

Dependencies

No dependencies exist for the Phantom resource.

Figure 6-2 Sample service group that includes a Phantom resource



Agent functions

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

Attribute

Table 6-6 Attribute

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

Resource type definition

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

Sample configurations

Configuration 1

```
Phantom boo (
)
```

Configuration 2

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

```
include "types.cf"

cluster PhantomCluster

system sysa

system sysb

group phantomgroup (
    SystemList = { sysa, sysb }
    AutoStartList = { sysa }
    Parallel = 1
    )

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

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

RemoteGroup agent

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

Note: RemoteGroup resources in a parallel service group are not supported.

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

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

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

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

Dependency

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

Agent functions

Online	Brings the remote service group online. See the "ControlMode" on page 199 for more information.
Offline	Takes the remote service group offline. See the "ControlMode" on page 199 for more information.
Monitor	Monitors the state of the remote service group. The true state of the remote service group is monitored only on the online node in the local cluster. See the "VCSSysName" on page 198.
Clean	If the RemoteGroup resource faults, the Clean function takes the remote service group offline. See the "ControlMode" on page 199 for more information.

State definitions

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

Attributes

Table 6-7 Required attributes

Required attribute	Description
IpAddress	The IP address or DNS name of a node in the remote cluster. The IP address can be either physical or virtual. When configuring a virtual IP address of a remote cluster, do not configure the IP resource as a part of the remote service group. Type and dimension: string-scalar Examples: "www.example.com" or "11.183.12.214"
Port	This is a required attribute when the remote cluster listens on a port other than the default value of 14141. See "Port" on page 201.
GroupName	The name of the service group on the remote cluster that you want the RemoteGroup agent to monitor or manage. Type and dimension: string-scalar Example: "DBGrp"
VCSSysName	You must set this attribute to either the VCS system name or the ANY value. ■ ANY The RemoteGroup resource goes online if the remote service group is online on any node in the remote cluster. ■ *VCSSysName* Use the name of a VCS system in a remote cluster where you want the remote service group to be online when the RemoteGroup resource goes online. Use this to establish a one-to-one mapping between the nodes of the local and remote clusters. Type and dimension: string-scalar Example: "vcssys1" or "ANY"

Table 6-7 Required attributes

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

Table 6-7 Required attributes

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

Table 6-8 Optional attributes

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

Table 6-9 Type-level attributes

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

Resource type definition

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

Chapter 7

Testing agents

This chapter contains the following agents:

About the testing agents

Use the testing agents to provide high availability for program support resources. These resources are useful for testing service groups.

ElifNone agent

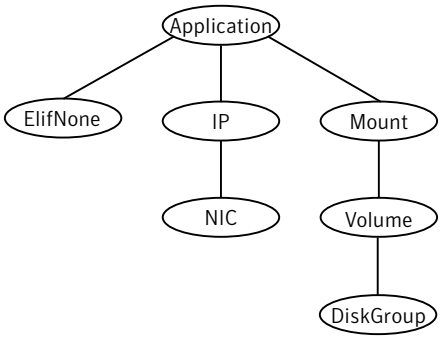
The ElifNone agent monitors a file. It checks for the file's absence.

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

Dependencies

No dependencies exist for the ElifNone resource.

Figure 7-1 Sample service group that includes an ElifNone resource



Agent function

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

State definitions

UNKNOWN — Indicates that the value of the PathName attribute does not contain a file name.

Attributes

Table 7-1 Required attribute

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

Resource type definition

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

Sample configuration

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

FileNone agent

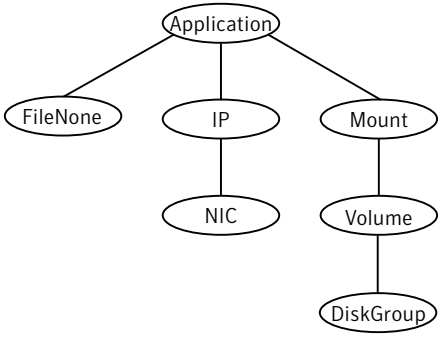
Monitors a file—checks for the file's existence.

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

Dependencies

No dependencies exist for the FileNone resource.

Figure 7-2 Sample service group that includes an FileNone resource



Agent functions

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

State definitions

UNKNOWN	Indicates that the value of the PathName attribute does not contain a file name.

Attribute

Table 7-2 Required attribute

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

Resource type definition

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

Sample configuration

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

FileOnOff agent

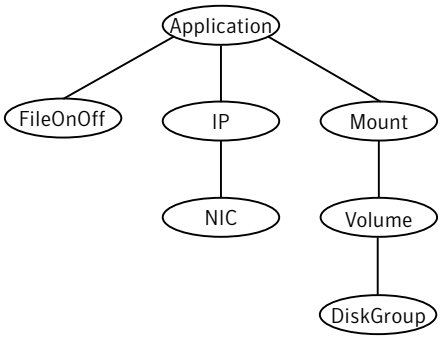
The FileOnOff agent creates, removes, and monitors files.

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

Dependencies

No dependencies exist for the FileOnOff resource.

Figure 7-3 Sample service group that includes a FileOnOff resource



Agent functions

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

State definitions

UNKNOWN — Indicates that the value of the PathName attribute does not contain a file name.

Attribute

Table 7-3 Required attribute

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

Resource type definition

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

Sample configuration

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

FileOnOnly agent

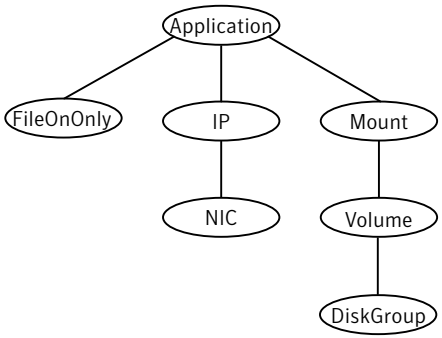
The FileOnOnly agent creates and monitors files.

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

Dependencies

No dependencies exist for the FileOnOnly resource.

Figure 7-4 Sample service group that includes a FileOnOnly resource



Agent functions

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

State definitions

UNKNOWN	Indicates that the value of the PathName attribute does not contain a file name.

Attribute

Table 7-4 Required attributes

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

Resource type definition

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

Sample configuration

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

Glossary

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

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

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

entry point
See agent function.

floating IP address
See virtual IP address.

logical IP address
Any IP address assigned to a NIC.

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

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

None operation
For example the NIC resource. Also called persistent resource, this resource is always on. This kind of resource has no online and offline scripts, and only monitors a resource.

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

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

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

test IP address

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

virtual IP address

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

Index

B

C

D

E

F

H

I

K

L

M

N

O

P

R

S

T

V