Veritas™ Cluster Server Bundled Agents Reference Guide

Linux

5.0

N18453H

Veritas Cluster ServerCluster Server Bundled Agents Reference Guide

Veritas Cluster Server 5.0

Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014
www.symantec.com

Third-party legal notices

Third-party software may be recommended, distributed, embedded, or bundled with this Symantec product. Such third-party software is licensed separately by its copyright holder. All third-party copyrights associated with this product are listed in the accompanying release notes.

Linux is a registered trademark of Linus Torvalds.

Technical support

For technical assistance, visit http://support.veritas.com and select phone or email support. Use the Knowledge Base search feature to access resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and our customer email notification service.

Contents

Chapter 4 File share 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 agents, see the VCS User's Guide.

Resources and their attributes

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

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

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

Modifying agents and their resources

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

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

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

Attributes

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

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

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.

When using volume sets, set StartVolumes and StopVolumes attributes of the DiskGroup resource that contains volume set to 1. If a file system is created on the volume set, use a Mount resource to mount the volume set.

The agent protects data integrity by disabling failover when data is being written to a volume in the disk group.

Virtual fire drill

The virtual 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 DiskGroup resources, the virtual fire drill checks for:

- The Veritas Volume Manager license
- Visibility from host for all disks in the diskgroup

For more information about using the virtual fire drill see the *VCS User's Guide*.

Dependency

This type of resource can depend on DiskReservation resources, provided that Dynamic Multipathing is *not* configured in Veritas Volume Manager.

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 off:
  ```
  #vxdg -g disk_group set autoimport=no
  ```
- 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 this 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 sliced
  FreeSize 35354136
  ```

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.
- 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 configured with Veritas Volume Manager. Type and dimension: string-scalar Example: "diskgroup1"
DiskGroupType	Different types of disk groups supported. SAN DG is supported only when the volume server is not centrally managed. Values are either: private or SAN. Type and dimension: string-scalar Example: "private"

Table 2-2 Optional attributes

Optional attributes	Description
MonitorReservation	If the value is 1, and SCSI-3 fencing is utilized, 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	If the DiskGroup resource is in a DISABLED state, uses I/O fencing, and has PanicSystemOnDGLoss set to 1, the system panics in the first monitor cycle. If the DiskGroup resource is in an ENABLED state, uses I/O fencing, has PanicSystemOnDGLoss set to 1, and fulfills the FaultOnMonitorTimeout attribute's time out number, the system panics. **Note:** System administrators may want to set a high value for FaultOnMonitorTimeout to increase system tolerance. Type and dimension: boolean-scalar Default: 1
StartVolumes	If value is 1, the DiskGroup online script starts all volumes belonging to that disk group after importing the group. Type and dimension: boolean-scalar Default: 1
StopVolumes	If value is 1, the DiskGroup offline script stops all volumes belonging to that disk group before deporting the group. Type and dimension: boolean-scalar Default: 1
TempUseFence	Do not use. For internal use only.

Resource type definition

```
type DiskGroup (
    static keylist SupportedActions = { "license.vfd", "disk.vfd",
    numdisks }
    static int NumThreads = 1
    static int OnlineRetryLimit = 1
    static str ArgList[] = { DiskGroup, StartVolumes, StopVolumes,
    MonitorOnly, MonitorReservation, PanicSystemOnDGLoss,
    tempUseFence, DiskGroupType }
    str DiskGroup
    boolean StartVolumes = 1
```

```
    boolean StopVolumes = 1
    boolean MonitorReservation = 0
    boolean PanicSystemOnDGLoss = 1
    temp str tempUseFence = INVALID
    str DiskGroupType = private
)
```

Configuring the Fiber Channel adapter

Most Fiber Channel (FC) drivers have a configurable parameter called “failover.” This configurable parameter is in the FC driver’s configuration file. This parameter is the number of seconds that the driver waits before transitioning a disk target from OFFLINE to FAILED. After the state becomes FAILED, the driver flushes all pending fiber channel commands back to the application with an error code. Symantec recommends that you use a non-zero value that is smaller than any of the MonitorTimeout values of the Disk Group resources, which avoids excessive waits for monitor timeouts.

Refer to the Fiber Channel adapter's configuration guide for further information.

Sample configurations

DiskGroup resource configuration

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

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

Example of a disk group resource in the Volume Serving mode.

```
SANVolume vNFS_SANVolume (
    Domain = testdom1
    SANDiskGroup = vsdg
    SANVolume = vsvol
    VolumeServer = "sysA.veritas.com"
)
```

DiskReservation agent

Reserves and monitors SCSI disks for a system, enabling a resource to go online on that system. This agent enables you to specify a list of raw disk devices, and reserve all or a percentage of accessible disks. The reservations prevent disk data corruption by restricting other nodes from accessing and writing to the disks. The DiskReservation agent supports all SCSI-II compliant disks.

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

The DiskReservation agent is not supported with dynamic multipathing software, such as Veritas DMP.

Note: The MonitorTimeout attribute's setting of 60 is adequate for up to three disks. When configuring the MonitorTimeout attribute for more than three disks, use the following formula:
Set MonitorTimeout to be equal or greater than 15 times the total number of disks. (MonitorTimeout >= 15 * Number of disks).
For example, if you have eight disks, MonitorTimeout is 120 or greater.

Agent functions

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

State definitions

- ONLINE
  Indicates that the number of reserved disks is greater than or equal to the percentage specified in the resource definition.

- OFFLINE
  Disks are not reserved.
- UNKNOWN
  Indicates that a problem exists with the configuration.

Attributes

Table 2-3 Required attributes

Required attribute	Description
Disks	A list of raw disk devices. Use the absolute or relative device path. The absolute or relative device path allows a maximum of 64 characters. The relative path is assumed to start from the `/dev` directory. The order of the disks in the list must be the same across all systems in the cluster, even if the same device has a different name on different systems. **Note:** You must change this attribute before bringing a resource online. An online device must be taken offline before altering this attribute because disk reservation occurs *during* the process of bringing a resource online. Type and dimension: string-vector Example: "c1t2d0s4"

Table 2-4 Optional attributes

Optional attribute	Description
FailFast	If enabled, FailFast causes the system to panic when a reservation conflict is detected, reducing the chance of further data corruption. Type and dimension: boolean-scalar Default: 0

Table 2-4 Optional attributes

Optional attribute	Description
Percentage	Minimum percentage of configured disks that can be reserved before a resource can go online. The percentage must be greater than or equal to 51, and less than or equal to 100. If the value specified is less than 51, the percentage is set to 51. If the value specified is greater than 100, the percentage is set to 100. Type and dimension: integer-scalar Default: 100
ProbeInterval	Alters the periodicity (in seconds) of the automatic probe function that checks the reservation status of the disks. The value must be greater than or equal to three, and less than or equal to 15. If the value specified is less than 3, the interval is set to 3. If the value specified is greater than 15, the interval is set to 15. A lower value for ProbeInterval specifies more frequent probes and provides for quicker discovery of reservation conflicts. Symantec recommends a value is between 3 and 8. Type and dimension: integer-scalar Default: 3

Resource type definition

```
type DiskReservation (
    static str ArgList[] = { Disks, FailFast, Percentage,
    ProbeInterval }
    str Disks[]
    boolean FailFast = 0
    int Percentage = 100
    int ProbeInterval = 3
)
```

Sample configurations

Configuration 1

In this example, the DiskReservation agent reserves a disk. The disk is mounted with the Veritas File System.

```
system sysA

system sysB

group groupx (
    SystemList = { sysA, sysB }
    AutoStartList = { sysA }
    )

    DiskReservation diskres1 (
        Disks = { "/dev/sdc" }
        FailFast = 1
        )

    Mount mount (
        MountPoint = "/mnt/tmp"
        BlockDevice = "/dev/sdc1"
        FSType = vxfs
        MountOpt = rw
        )

    mount requires diskres1

    // resource dependency tree
    //
    //  group groupx
    //  {
    //  Mount mount
    //      {
    //       DiskReservation diskres1
    //       }
    //  }
```

Configuration 2

In this example, the DiskReservation agent reserves several disks. The disk group defined on these disks is imported only if the system can reserve the disks.

Volumes can be enabled and mounted on the Disk Group. Refer to the Volume agent for a sample configuration.

```
group groupy (
    SystemList = { sysA, sysB }
    AutoStartList = { sysA }
    )

    DiskGroup resdg (
        DiskGroup = resdg
        )

    DiskReservation diskres2 (
        Disks = { "/dev/sdc", "/dev/sdd", "/dev/sde", "/dev/
        sdf", "/dev/sdg" }
        ProbeInterval = 5
        Percentage = 60
        )

    resdg requires diskres2

    // resource dependency tree
    //
    //  group groupy
    //  {
    //  DiskGroup resdg
    //     {
    //     DiskReservation diskres2
    //     }
    //  }
```

Volume agent

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

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

Dependency

Volume resources depend on DiskGroup resources.

Agent functions

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

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 stops unexpectedly.
- UNKNOWN
  Indicates that the agent could not determine the state of the resource or that the resource attributes are invalid.

Attributes

Table 2-5 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. Type and dimension: string-scalar Example: "vol3"

Resource type definition

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

Sample configurations

Configuration

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

Configuration

In this example, the DiskReservation resource is used to verify that disks are available only to one system. The volumes on the disk groups that are imported are started if the reservation is confirmed. The volumes can then be mounted at a mount point.

```
group groupy (
    SystemList = { sysA, sysB }
    AutoStartList = { sysA }
```

```
)

DiskGroup resdg (
    DiskGroup = resdg
    )

DiskReservation diskres2 (
    Disks = { "/dev/sdc", "/dev/sdd", "/dev/sde", "/dev/sdf",
    "/dev/sdg" }
    ProbeInterval = 5
    Percentage = 60
    )

Mount mountvol (
    BlockDevice = "/dev/vx/dsk/resdg/resvol"
    MountPoint = "/share"
    FSType = vxfs
    MountOpt = rw
    )

Volume resdg_resvol (
    DiskGroup = resdg
    Volume = resvol
    )

mountvol requires resdg_resvol
resdg requires diskres2
resdg_resvol requires resdg

// resource dependency tree
//
//  group groupy
//  {
//  Mount mountvol
//      {
//      Volume resdg_resvol
//          {
//          DiskGroup resdg
//              {
//              DiskReservation diskres2
//              }
//          }
//      }
//  }
```

LVMLogicalVolume agent

Brings online, takes offline, and monitors a Logical Volume Manager (LVM2) volume. This agent uses LVM2 commands.

Dependency

LVMLogicalVolume resources depend on LVMVolumeGroup resources.

Agent functions

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

State definitions

- ONLINE
  Indicates that the specified volume is started and that I/O is permitted.
- OFFLINE
  Indicates that the specified volume is not started—and I/O is not permitted.
- UNKNOWN
  Indicates that the agent could not determine the state of the resource or that the resource attributes are invalid.

Attributes

Table 2-6 Required attributes

Required attribute	Description
LogicalVolume	Name of the volume configured with Logical Volume Manager (LVM2). Type and dimension: string-scalar Example: "volume1"
VolumeGroup	Name of the volume group configured with Logical Volume Manager (LVM2). which contains the volume. Type and dimension: string-scalar Example: "volumegroup1"

Resource type definition

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

Sample configurations

Configuration 1

In this example, /dev/sdc and /dev/sdd are the disks where the volume group testvg_1 is created.

```
LVMLogicalVolume lvol1 (
    LogicalVolume = testvol_1
    VolumeGroup = testvg_1
)

LVMVolumeGroup lvg1 (
    VolumeGroup = testvg_1
)

DiskReservation dr1 (
    Disks = { "/dev/sdc", "/dev/sdd" }
)
```

```
lvol1 requires lvg1
lvg1 requires dr1
```

Configuration 2

In this example, you use the DiskReservation resource to verify that disks are available only to one system. The LVM2 logical volumes on the LVM2 volume groups that are imported are started if the reservation is confirmed. The logical volumes can then be mounted at a mount point.

```
DiskReservation dr_cde (
    Disks = { "/dev/sdc", "/dev/sdd", "/dev/sde" }
    )

    Mount mnt_lvmvol01 (
        MountPoint = "/mnt/lvmvol01"
        BlockDevice = "/dev/mapper/lvmvg01-lvmvol01"
        FSType = "reiserfs"
        FsckOpt = "-y"
        )

    LVMLogicalVolume lvmvol01 (
        LogicalVolume = lvmvol01
        VolumeGroup = lvmvg01
        )

    LVMVolumeGroup lvmvg01 (
        VolumeGroup = lvmvg01
        )

    mnt_lvmvol01 requires lvmvol01
    lvmvol01 rquires lvmvg01
    lvmvg01 requires dr_cde
```

LVMVolumeGroup agent

Brings online, takes offline, and monitors a Logical Volume Manager (LVM2) volume group. This agent uses LVM2 commands.

Dependency

LVMVolumeGroup resources depends on DiskReservation resources. Without DiskReservation resources the LVMVolumeGroup can not function.

Agent functions

- Online
  Imports the volume group using the `vgimport` command.
- Offline
  Exports the volume group using the `vgexport` command.
- Monitor
  Determines if the volume group is online or offline using the `vgdisplay` command.
- Clean
  Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

State definitions

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

Attributes

Table 2-7 Required attributes

Required attribute	Description
VolumeGroup	Name of the volume group configured with Logical Volume Manager (LVM2) that contains the volume. Type and dimension: string-scalar Example: "volumegroup1"

Table 2-8 Optional attributes

Optional attribute	Description
StartVolumes	If set to 1, the LVMVolumeGroup online script starts all volumes belonging to that volume group after importing the group. Type and dimension: boolean-scalar Default: 0

Resource type definition

```
type LVMVolumeGroup (
    static str ArgList[] = { VolumeGroup, StartVolumes }
    str VolumeGroup
    boolean StartVolumes = 0
)
```

Sample configurations

Linux configuration 1

In this example, /dev/sdc and /dev/sdd are the disks where the volume group testvg_1 is created.

```
LVMVolumeGroup lvg1 (
    VolumeGroup = testvg_1
)

DiskReservation dr1 (
```

```
    Disks = { "/dev/sdc", "/dev/sdd" }
)

lvg1 requires dr1
```

Linux configuration 2

In this example, the DiskReservation resource is used to verify that disks are available only to one system. All LVM2 logical volumes on the LVM2 volume groups that are imported are started if the reservation is confirmed. You can then mount the logical volumes at a mount point.

```
DiskReservation dr_cde (
    Disks = { "/dev/sdc", "/dev/sdd", "/dev/sde" }
    )

    Mount mnt_lvmvol01 (
        MountPoint = "/mnt/lvmvol01"
        BlockDevice = "/dev/mapper/lvmvg01-lvmvol01"
        FSType = "reiserfs"
        FsckOpt = "-y"
        )

    LVMVolumeGroup lvmvg01 (
        VolumeGroup = lvmvg01
        StartVolumes = 1
        )

    mnt_lvmvol01 requires lvmvg01
    lvmvg01 requires dr_cde
```

Mount agent

Use this agent to bring online, take offline, and monitor a file system or NFS client mount point.

Virtual fire drill

The virtual 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 virtual fire drill checks for:

- The existence of the mount directory
- The correct filesystem mounted at the specified mount directory

For more information about using the virtual fire drill see the *VCS User's Guide*.

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 to remount the block device.
  If file system type is NFS, agent mounts the remote NFS 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:
  ```
  df -h 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%
```

State definitions

- ONLINE

  Indicates that the filesystem is properly mounted on the given mount point.

- OFFLINE

  Indicates that the filesystem is not mounted properly mounted on the mountpoint.

- FAULTED

  Indicates that the filesystem got unmounted unexpectedly.

- UNKNOWN

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

Attributes

Table 2-9 Required attributes

Required attribute	Description
BlockDevice	Block device for mount point. Type and dimension: string-scalar Examples: /dev/sdc1, /dev/vx/dsk/dg/volume If the device is LVM2 volume, then you must specify the BlockDevice as: /dev/*mapper/volume-group-logical-volume* LVM2 volume example: /dev/mapper/lvg-lvolume If the file system type is NFS, then specify the BlockDevice as: server:/path/to/share NFS device example: vcslnx1.veritas.com:/usr/share1
FsckOpt	Options for `fsck` command. Attribute value can contain -y or -n. For file systems (non-NFS mounts), the FsckOpt attribute is mandatory. If the mount process fails, `fsck` runs with the specified options before attempting to remount the block device. Type and dimension: string-scalar VxFS example: `-y -o full`
MountPoint	Directory for mount point. Type and dimension: string-scalar Example: "/mnt/apache1"

Table 2-10 Optional attributes

Optional attribute	Description
CkptUmount	If set to 1, this attribute automatically unmounts VxFS checkpoints when the file system is unmounted. If set to 0, and checkpoints are mounted, then failover does not occur. Type and dimension: boolean-scalar Default: 1
FSType	Type of file system. Supports vxfs, bind, ext2, ext3, nfs, or reiserfs. If set to NFS, the agent always mounts the file system with the “soft,intr” option. Type and dimension: string-scalar Default: vxfs
MountOpt	Options for the `mount` command. Refer to the `mount` manual page for more information. Do not specify `-o` in the MountOpt field. The agent uses this option only when bringing a Mount resource online. If set to nfs, the agent always mounts the file system with the “soft,intr” option. Type and dimension: string-scalar Example: "rw"
SecondLevelMonitor	This attribute is only applicable to NFS client mounts. If set to 1, this attribute enables detailed monitoring of a NFS mounted file system. Type and dimension: boolean-scalar Default: 0

Table 2-10 Optional attributes

Optional attribute	Description
SecondLevelTimeout	This attribute is only applicable for a NFS client mount. This is the timeout (in seconds) for the SecondLevelMonitor/Detail monitoring of NFS Mounts. Type and dimension: integer-scalar Default: 30
SnapUmount	If set to 1, this attribute automatically unmounts VxFS snapshots when the file system is unmounted. If set to 0 and snapshots are mounted, then failover does not occur. Type and dimension: boolean-scalar Default: 0

Resource type definition

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

Sample configurations

Configuration

```
Mount MountSCSI1 (
    MountPoint= "/scsi1"
    BlockDevice = "/dev/sda1"
```

```
        FSType = ext2
        MountOpt = rw
        FsckOpt = "-y"
    )
```

SANVolume agent

Use this agent as a resource to control access to a SAN volume, and to monitor the health of a SAN volume. You can configure the agent as part of a VCS service group.

The SAN volumes must reside on storage arrays that support SCSI-3 persistent reservations.

Note: Storage Foundation Volume Server (SF Volume Server) is a separately licensed feature of Veritas Storage Foundation™ by Symantec. An SF Volume Server license is currently available only through the Symantec customer access program. For information about participating in the access program and obtaining an SF Volume Server license, visit the following Symantec website: http://cap.symantec.com

---

Agent functions

- Online
  Attaches the SAN volume to the volume client host, and creates a device node for the SAN volume on the volume client host.
- Offline
  Unattaches a SAN volume. It deletes the device node for the already attached SAN volume to the volume client host.
- Clean
  Forcibly detaches the SAN volume from a volume client.
- Monitor
  Checks the state of the SAN volume on a volume client. It checks the health of the SAN volume and determines whether it is online or offline.

State definitions

- ONLINE
  Indicates that state of the SAN volume is attached.
- OFFLINE
  Indicates that the SAN volume is unattached.
- 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
DiskGroup	The name of the SAN disk group that contains the volume. Type and dimension: string-scalar Example: "dg1"
Domain	The name of the storage domain that the SAN volume belongs to. Type and dimension: string-scalar Example: "domain1"
SANVolume	The name of the SAN volume Type and dimension: string-scalar Example: "sanvol_1"
VolumeServer	The name of the SAN volume server. ■ If the volume server is not centrally managed, then this is required. If the volume server is made highly available using VCS, then the name of the volume server should be a virtual IP address or the host name associated with the virtual IP address. ■ For a centrally managed volume server, then this attribute is not required. Type and dimension: string-scalar Example: "myserver.veritas.com"

Table 2-12 Optional attributes

Optional attribute	Description
ExclusiveUse	ExclusiveUse enforces volume to be opened by only one node in the cluster at a time. Type and dimension: boolean-scalar Default: 1
Preempt	Preempt enforces an exclusive attach of the volume by a node in the cluster. Type and dimension: integer-scalar Default: 1
AccessPolicy	The access policy for the volume: {RDONLY | RDWR} Type and dimension: string-scalar Default: RDWR

Resource type definition

```
type SANVolume (
    static int OnlineRetryLimit = 4
    static str ArgList[] = { SANVolume, SANDiskGroup, VolumeServer,
    Domain, ExclusiveUse, Preempt, AccessPolicy }
    str Domain
    str SANDiskGroup
    str SANVolume
    str VolumeServer
    boolean ExclusiveUse = 1
    boolean Preempt = 1
    str AccessPolicy = RDWR
)
```

Sample configuration

This example shows all the required attributes.

```
SANVolume svol (
    SANDiskGroup = vsdg
    SANVolume = vsvol
    VolumeServer = "sysA.veritas.com"
)
```

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

IPMultiNIC and MultiNICA agents

The IPMultiNIC and MultiNICA agents:

- Operate in two modes:
  - IP Conservation (IPC) Mode, which uses fewer IP addresses
  - Performance Mode (PM), which provides faster failover, but uses more IP addresses
- Monitor single or multiple NICs
- Check the backup NICs at fail over
- Use the original base IP address when failing over
- Have only one active NIC at a time

802.1Q trunking

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

The underlying utility to manage 802.1Q trunk interfaces is vconfig. For example, you can create a trunk interface on the physical interface:

```
# vconfig add eth2 10
```

This creates a trunk interface called eth2.10 in the default configuration. In this case, the physical NIC eth2 must be connected to a trunk port on the switch. You can now use eth2.10 like a regular physical NIC in a NIC, IP, and MultiNICA resource configuration. You can remove it with the following comand.

```
# vconfig rem eth2.10
```

VCS does not create nor remove trunk interfaces. The administrator should set up the trunking as per the operating system vendor's documentation rather than using vconfig directly.

IP agent

Manages the process of configuring a virtual IP address and its subnet mask on an interface. The interface must be enabled with a physical (or administrative) base IP address before you can assign it a virtual IP address. The virtual IP address must not be in use.

Virtual fire drill

The virtual 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 IP resources, the virtual fire drill checks for the existence of a route to the IP from the specified NIC.

For more information about using the virtual fire drill see the *VCS User's Guide*.

Dependency

IP resources depend on NIC resources.

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.
  Linux: Sends out a gratuitous ARP.
- Offline
  Brings down the IP address 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 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 attribute	Description
Address	A virtual IP address, different from the base IP address, which is associated with the interface—the address you specify must not be the same as the configured physical IP address. Type and dimension: string-scalar Example: "192.203.47.61"
Device	The name of the NIC device associated with the IP address. Requires the device name without an alias. Type and dimension: string-scalar Example: Specify eth0 to assign the IP address to the next available alias. Use the `ifconfig -a` command to display a list of NICs that are up and the IP addresses assigned to each NIC.

Table 3-2 Optional attributes

Optional attribute	Description
NetMask	Subnet mask associated with the IP address. Specify the value of NetMask in decimal (base 10). If Netmask is not specified, the agent uses the operating system's default netmask. Type and dimension: string-scalar Example: "255.255.255.0"
Options	Options for the `ifconfig` command. Type and dimension: string-scalar Example: "broadcast 172.20.9.255"

Resource type definition

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

Sample configurations

Configuration 1

```
IP          IP_192_203_47_61 (
    Device = eth0
    Address = "192.203.47.61"
    )
```

Configuration using specified NetMask

```
IP          IP_192_203_47_61 (
    Device = eth0
    Address = "192.203.47.61"
    NetMask = "255.255.248.0"
    )
```

NIC agent

Monitors the configured NIC. If a network link fails, or if a problem arises with the NIC, the resource is marked FAULTED.

Some NICs maintain their connection status in a hardware register. For NICs that maintain their connection status, the agent uses MII to determine the status of the NIC resource. For NICs that do not maintain their connection status, the agent uses a ping or a broadcast to determine the status of the resource.

Virtual fire drill

The virtual 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 NIC resources, the virtual fire drill checks for the existence of the NIC on the host.

For more information about using the virtual fire drill see the *VCS User's Guide*.

Bonded network interfaces

The NIC agent now supports using bonded network interfaces.

See "Monitoring bonded NICs" on page 57.

Agent functions

- Monitor

  If the NIC maintains its connection status, the agent uses MII to determine the status of the resource.

  If the NIC does not maintain its connection status, the agent verifies that the NIC is configured and sends a ping to all the hosts listed in the NetworkHosts attribute. If the ping test is successful, it marks the NIC resource ONLINE.

  If the NetworkHosts list is empty, or the ping test fails, the agent counts the number of packets received by the NIC and compares the count with a previously stored value. If the packet count increases, the resource is marked ONLINE. If the count remains unchanged, the agent sends a ping to the broadcast address of the device to generate traffic on the network.

  The agent counts the number of packets received by the NIC before and after the broadcast. If the count increases, the resource is marked ONLINE. If the count remains the same or decreases over a period of five broadcast cycles, the resource is marked OFFLINE.

State definitions

- ONLINE
  Indicates that the NIC resource is working.
- OFFLINE
  The NIC resource can go OFFLINE if the NIC it represents has failed or is unavailable.
- 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. Use the `ifconfig -a` command to list all network adapters and the IP addresses assigned to each NIC. Type and dimension: string-scalar Example: "eth0" or "eth1"

Table 3-4 Optional attributes

Optional attribute	Description
Mii	Flag that defines whether the NIC maintains its connection status. If this flag is set to 1, the agent uses MII hardware registers, instead of the ping and packet count method, to determine the health of the network card. If the flag is set to 0, the agent does not use Mii to monitor the status of the NIC. Type and dimension: integer-scalar Default: 1
NetworkHosts	List of hosts on the network that receive pings to determine the state of the NIC. Specify the IP address of the host—not the host name. The specified hosts must be pingable: ■ from all the AppNodes specified in the SystemList attribute for the service group to which the resource belongs ■ through all the device specified in the Device attribute The command to ping the host (hostip) via a NIC device (nicdev) is: **# ping -I *nicdev hostip*** If more than one network host is listed, the monitor returns ONLINE if the ping test is successful with at least one of the hosts. Type and dimension: string-vector
PingOptimize	Flag that defines whether the agent sends a broadcast ping before retrieving the received packet statistics. This attribute is used when Mii is not set and no network hosts are specified. If this flag is set to 1, the agent will retrieve received packet statistics from the netstat command and compare them with previously stored values. The agent sends a broadcast ping to the network only if the packet count remains unchanged. If this flag is set to 0, the agent sends a broadcast ping before checking the network statistics. Type and dimension: integer-scalar Default: 1

Resource type definition

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

Monitoring bonded NICs

The NIC agent can monitor the network interfaces (bond0, bond1, etc.) that the bonding driver exports. Refer to operating system vendor documentation to set up the bonds and to configure your system to load the bonding driver correctly.

For monitoring a bond interface, the two important settings are:

- The value of the miimon parameter, which you set while loading the bonding driver. miimon is a parameter to the bonding module and has a default setting of 0.
- The value of the Mii attribute (Mii) of the NIC resource, which you set at runtime. Mii is an attribute of the NIC resource and has a default setting of 0.

Setting Mii and miimon

For the following cases, the name of the monitored bond interface is B. If you do not use one of the following cases to set up bonding, the bonding driver can potentially provide incorrect health status, which can result in VCS failing to fault the resource appropriately.

Case 1

Accept defaults—miimon is 0 and Mii is 1. Each of B's slaves must support the netif_carrier_ok in-kernel call.

Case 2

When you set miimon to anything except 0 (miimon!=0) and Mii to 1, both the hardware and the drivers of each of B's slaves must support the MII-based health monitoring.

Case 3

When you set Mii to 0, the NIC agent uses ping, which each card supports. In this case, the miimon setting is irrelevant.

Sample configurations

Configuration for using Mii

If the NIC does not respond to Mii, the agent uses network statistics to monitor the device.

```
NIC groupx_eth0 (
    Device = eth0
    Mii = 1
    PingOptimize = 1
    )
```

Configuration for using network hosts

```
NIC groupx_eth0 (
    Device = eth0
    NetworkHosts = { "166.93.2.1", "166.99.1.2" }
    )
```

IPMultiNIC agent

Manages the virtual IP address configured as an alias on one interface of a MultiNICA resource. If the interface faults, the agent works with the MultiNICA resource to fail over to a backup NIC. If multiple service groups have IPMultiNICs associated with the same MultiNICA resource, only one group has the MultiNICA resource. The other groups have Proxy resources pointing to it.

Dependency

IPMultiNIC resources depend on MultiNICA resources.

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.

State definitions

- ONLINE
  Indicates that the specified IP address is assigned to the device.
  Sends out a gratuitous ARP.
- 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 may be due to an incorrect configuration.

Attributes

Table 3-5 Required attributes

Required attribute	Description
Address	Virtual IP address that is assigned to the active NIC. Type and dimension: string-scalar Example: "10.128.10.14"
MultiNICAResName	Name of associated MultiNICA resource that determines the active NIC. Type and dimension: string-scalar Example: "MultiNICA_grp1"

Table 3-6 Optional attributes

Optional attribute	Description
NetMask	Netmask for the virtual IP address. Specify the value of NetMask in decimal (base 10). Type and dimension: string-scalar Example: "255.255.255.0"
Options	The `ifconfig` command options for the virtual IP address. Type and dimension: string-scalar Example: "mtu m"

Resource type definition

```
type IPMultiNIC (
    static int MonitorTimeout = 200
    static int OfflineMonitorInterval = 120
    static int ToleranceLimit = 2
    static str ArgList[] = { Address, NetMask, MultiNICAResName,
    Options, "MultiNICAResName:Probed" }
```

```
    str Address
    str MultiNICAResName
    str NetMask
    str Options
)
```

Sample configuration: IPMultiNIC and MultiNICA

```
cluster foo (
    UserNames = { admin = "cDRpdxPmHpzS." }
    CounterInterval = 5
    )
system vcslx3 (
    )
system vcslx4 (
    )
group grp1 (
    SystemList = { vcslx3 = 1, vcslx4 = 2 }
    )

    IPMultiNIC ip1 (
        Address = "192.123.10.177"
        MultiNICAResName = mnic
        NetMask = "255.255.248.0"
        )

    MultiNICA mnic (
        Device @vcslx3 = { eth0 = "192.123.10.127", eth1 =
"192.123.11.127" }
        Device @vcslx4 = { eth0 = "192.123.10.128", eth2 =
"192.123.11.128" }
        NetMask = "255.255.248.0"
        NetworkHosts = { "192.123.10.129", "192.123.10.130" }
        )

    ip1 requires mnic


    // resource dependency tree
    //
    //          group grp1
    //          {
    //          IPMultiNIC ip1
    //               {
    //               MultiNICA mnic
    //               }
    //          }
```

MultiNICA notes

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

MultiNICA agent

Represents a set of network interfaces, and provides failover capabilities between them. The IPMultiNIC agent depends upon the MultiNICA agent to select the most preferred NIC on the system. IPMultiNIC brings the virtual IP online or offline. However, if the MultiNICA resource changes its active device, the MultiNICA agent handles the shifting of IP addresses.

If a NIC on a system fails, the MultiNICA agent selects another active NIC, and the virtual IP address shifts to the newly selected active NIC. Only in a case where all the NICs that form a MultiNICA agent fail, does the virtual IP address shift to another system.

If you associate an interface with a MultiNICA resource, do not associate it with any other MultiNICA 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, and
- Proxy resources that point to the MultiNICA resource in the other service groups.

The MultiNICA agent can operate in two modes “IP Conservation Mode (ICM)” on page 63 and “Performance Mode (PM)” on page 64. With sufficient IP addresses, use PM.

IP Conservation Mode (ICM)

Requires fewer IP addresses than Performance Mode, but provides slower failover.

Configuration

When a MultiNICA resource controls all the NICs of a cluster node, the NICs must have the same base IP address. This IP address must be unique, and cannot appear on any other NIC on any other node. You do not need to enable the base IP addresses beforehand. This mode does not support failing back the NIC, see the optional Failback attribute.

Operation

When you specify all the NICs with the same base IP address, the agent runs in ICM. It enables the base IP address on the active NIC.

In case of a failover, it moves the base IP address to the new active NIC. It also moves all the virtual IP addresses configured on that NIC and it attempts to find the first working NIC in the order of priority.

Performance Mode (PM)

Requires more IP addresses than ICM, but provides faster failover. You do not have to spend time enabling and disabling base IP addresses and reinstating lost routes, thus no resultant service disruption occurs.

Configuration

The MultiNICA agent controls each NIC, and each NIC must have a unique base IP address. This IP address cannot appear on any other NIC on the same node or any other node. When you configure a single NIC under a MultiNICA resource, the MultiNICA agent uses PM. The base IP addresses have to be enabled on each NIC under MultiNICA control before starting VCS and handing over the management of the NICs to the agent.

Operation

The agent uses this mode when all NICs under the MultiNICA agent have separate base IP addresses specified. The mode requires that you enable the base IP addresses before starting VCS. When a NIC goes down, the agent migrates only virtual IP addresses.

In this mode, you can set the Failback attribute to 1 or 0:

- If you set the Failback attribute to 1, in each monitor cycle the agent checks to see if a preferred NIC is up. If the NIC is up, it selects that NIC as the active NIC and moves the virtual IP addresses to the preferred NIC.
- If you set the Failback attribute to 0, the agent selects a new active NIC only if the current active NIC fails. It selects the new active NIC in the order of priority.

Agent function

- Monitor
  Uses Medium Independent Interface (MII) to request the device status. If the hardware does not respond, the agent sends a ping to the hosts listed in the NetworkHosts attribute. If the ping test fails, the agent checks for activity on a configured interface by sampling input packets received on that interface. If the agent does not detect activity, it forces activity by sending out a broadcast ping. If the agent does not receive a network reply, it migrates to the most suitable next interface.

Attributes

Table 3-7 Required attributes

Required attribute	Description
Device	List of devices and associated base IP addresses. This attribute must be specified separately for each system in the SystemList. You must specify the devices in the list in the order of priority. The first device that the agent determines is “up” becomes the active device, to which the agent assigns a corresponding IP address. In the IP Conservation Mode (ICM), if all the NICs configured in the Device attribute are down, the MultiNICA agent faults the resource after a 2-3 minute interval. This delay occurs because the MultiNICA agent tests the failed NIC several times before marking the resource offline. Messages recorded in the engine log during failover provide a detailed description of the events that take place during failover, find the engine log in /var/VRTSvcs/log/engine_A.log. Type and dimension: string-association Example: Device@vcslinux1={ eth1 = "10.212.100.178", eth2 = "10.212.102.178" } Device@vcslinux2 ={ eth2 = "10.212.100.179", eth3 = "10.212.102.179" }
NetMask	Netmask associated with the base IP address. The value must be specified in decimal (base 10). Type and dimension: string-scalar

Table 3-8 Optional attributes

Optional attribute	Description
Failback	In the "Performance Mode (PM)," this attribute determines if the active NIC should be changed to a preferred NIC, even though the current NIC is healthy. If operating in the ICM mode, change the value to 0. Type and dimension: boolean-scalar Default: 1
NetworkHosts	List of hosts on the network that receive pings to determine the state of the NICs. Specify the IP address of the host, not the host name. Include hosts that can be reached by all the NICs in the Device list. If more than one network host is listed, monitor returns ONLINE if the ping test is successful with at least one of the hosts. Type and dimension: string-vector
Options	The ifconfig options involved while assigning the base IP address to the active device. Type and dimension: string-scalar Example: "broadcast 10.212.100.255"
PingOptimize	Determines whether or not a broadcast ping is sent prior to checking network statistics, which are used to determine the state of the NIC (if MII is not supported and the ping to NetworkHosts does not confirm the NIC is up.) A value of 1 indicates a broadcast ping does not occur, a value of 0 indicates a broadcast ping occurs. Type and dimension: integer-scalar Default: 1

Table 3-8 Optional attributes

Optional attribute	Description
RouteOptions	Assignment of a base IP address to a device is followed by a route `add` command with the options specified for this attribute. RouteOptions are applicable only when configuring the local host as the default gateway. No routes are added if this string is set to NULL. Type and dimension: string-scalar Example: "default 166.98.16.103"

Resource type definition

```
type MultiNICA (
    static int MonitorTimeout = 240
    static str ArgList[] = { Device, NetMask, Options,
    RouteOptions,PingOptimize, MonitorOnly, NetworkHosts,
    Failback }
    static str Operations = None
    str Device{}
    str NetMask
    str Options
    str RouteOptions
    int PingOptimize = 1
    str NetworkHosts[]
    boolean Failback = 1
)
```

Sample configurations

MultiNICA and IPMultiNIC Performance Mode configuration

In this example, two systems (vcslx3 and vcslx4) each have a pair of network interfaces (eth0 and eth1, eth0 and eth2). These interfaces have different physical IP addresses and the agent behaves in Performance Mode (PM).

The MultiNICA resource fails over only the logical IP address to the backup NIC in the event of a failure. The resource ip1 has the Address attribute, which contains the logical IP address. In the event of a NIC failure on vcslx3, the logical IP address fails over from eth0 to eth1. In the event that eth1 fails—the address fails back to eth0—as long as eth0 is reconnected.

However, if both the NICs on vcslx3 are disconnected, the MultiNICA and IPMultiNIC resources work in tandem to fault the group on vcslx3. The entire group fails over to vcslx4.

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

See “IPMultiNIC agent” on page 59.

```
cluster foo (
    UserNames = { admin = "cDRpdxPmHpzS." }
    CounterInterval = 5
    )

    system vcslx3 (
    )
    system vcslx4 (
    )

group grp1 (
    SystemList = { vcslx3 = 1, vcslx4 = 2 }
    )

IPMultiNIC ip1 (
    Address = "192.123.10.177"
    MultiNICAResName = mnic
    NetMask = "255.255.248.0"
    )

MultiNICA mnic (
    Device @vcslx3 = { eth0 = "192.123.10.127", eth1 =
    "192.123.11.127" }
    Device @vcslx4 = { eth0 = "192.123.10.128", eth2 =
    "192.123.11.128" }
    NetMask = "255.255.248.0"
    NetworkHosts = { "192.123.10.129", "192.123.10.130" }
    )

ip1 requires mnic

    // resource dependency tree
    //
    // group grp1
    //          {
    // IPMultiNIC ip1
    //          {
    //          MultiNICA mnic
    //          }
    //     }
```

MultiNICA and IPMultiNIC IP Conservation Mode Configuration

In this example, two systems (vcslx3 and vcslx4) each have a pair of network interfaces (eth0 and eth1, eth0 and eth2). These interfaces have a common physical IP address and the agent behaves in IP Conservation Mode (ICM).

The MultiNICA resource fails over both the physical IP and the logical IP addresses to the backup NIC in the event of a failure. The resource ip1 has the Address attribute, which contains the logical IP address. In the event of a NIC failure on vcslx3, the IP addresses fail over from eth0 to eth1. In the event that eth1 fails—the addresses fail back to eth0—if eth0 is reconnected.

However, if both the NICs on vcslx3 are disconnected, the MultiNICA and IPMultiNIC resources work in tandem to fault the group on vcslx3. The entire group fails over to vcslx4.

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

See “IPMultiNIC agent” on page 59.

```
cluster foo (
    UserNames = { admin = "cDRpdxPmHpzS." }
    CounterInterval = 5
    )

system vcslx3 (
    )
system vcslx4 (
    )

group grp1 (
    SystemList = { vcslx3 = 1, vcslx4 = 2 }
    )
IPMultiNIC ip1 (
    Address = "192.123.10.177"
    MultiNICAResName = mnic
    NetMask = "255.255.248.0"
    )

MultiNICA mnic (
    Device @vcslx3 = { eth0 = "192.123.10.127", eth1 =
    "192.123.10.127" }
    Device @vcslx4 = { eth0 = "192.123.10.128", eth2 =
    "192.123.10.128" }
    NetMask = "255.255.248.0"
    NetworkHosts = { "192.123.10.129", "192.123.10.130" }
    Failback = 0

    )
```

```
ip1 requires mnic

    // resource dependency tree
    //
    // group grp1
    //          {
    // IPMultiNIC ip1
    //          {
    //          MultiNICA mnic
    //          }
    //     }
```

DNS agent

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

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.

Agent functions

- Online
  Queries the authoritative name server of the domain for CNAME records and updates the CNAME record on the name server with the specified alias to canonical name mapping. Adds a new CNAME record if a related record is not found. Creates an Online lock file if the Online function was successful.
- Offline
  Removes the Online lock file, which the Online agent function created.
- Monitor
  If the Online lock file exists, the Monitor function queries the name servers for the CNAME record for the alias. It reports back ONLINE if the response from at least one of the name servers contains the same canonical name associated with the alias in the Hostname attribute. If no servers return the appropriate name, the monitor reports the resource as OFFLINE.
- Clean
  Removes the Online lock file, if it exists.
- Open
  Removes the Online lock file if the Online lock file exists, and the CNAME record on the name server does not contain the expected alias or canonical name mapping.

State definitions

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

Attributes

Table 3-9 Required attributes

Required attribute	Description
Alias	A string representing the alias to the canonical name. Type and dimension: string-scalar Example: "www" Where www is the alias to the canonical name mtv.veritas.com.
Domain	A string representing the domain name. Type and dimension: string-scalar Example: "veritas.com"
Hostname	A string representing canonical name of a system. Type and dimension: string-scalar Example: "mtv.veritas.com"
TTL	A non-zero integer representing the Time To Live (TTL) value, in seconds, for the DNS entries in the zone you are updating. A lower value means more hits on your DNS server, while a higher value means more time for your clients to learn about changes. Type and dimension: integer-scalar Default: 86400 Example: "3600"

Table 3-10 Optional attributes

Optional attribute	Description
StealthMasters	The list of primary master name servers in the domain. Optional if the zone's name server record lists the primary master name server. 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 is not listed in that zone's name server records. 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/Kveritas.com.+157+00000.private

Resource type definition

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

Online query

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

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

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

Monitor scenarios

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

Table 3-11 Monitor scenarios for the Online lock file

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

Note: The DNS agent supports BIND version 8 and above.

Sample web server configuration

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

Sample DNS configuration

```
DNS www (
        Domain = "example.com"
        Alias = www
```

```
Hostname = virtual1
)
```

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

- Linux

```
www   CNAME   wwwvirtual1
```

All DNS lookups for www.example.com resolve to www.virtual1.example.com.

DNS agent prerequisites

For the DNS agent to work correctly, set up the primary and the stealth masters to accept and correctly process updates. You can test this via the `nsupdate` command that ships with the operating system—refer to the `nsupdate` manpage for usage. By default both SuSE and Red Hat install with SELinux enabled and configured to disallow the DNS server process named, to modify the on-file DNS database—the zone files. You must set up SELinux on the DNS server systems to allow the named process to update. Refer to the operating system documentation for more information.

Secure DNS update

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

Setting up secure updates using TSIG keys for Linux

In the following example, the domain is veritas.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 512 -n HOST veritas.com.
    Kveritas.com.+157+00000
```

2. Open the Kveritas.com.+157+00000.key file. After running the `cat` command, the contents of the file resembles:

```
# cat Kveritas.com.+157+00000.key
    veritas.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 veritas.com. {
    algorithm hmac-md5;
    secret "+Cdjlkef9ZTSeixERZ433Q==";
};
```

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

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

```
allow-updates { key veritas.com. ; } ;
```

6 Save and restart the named process.

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

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

```
DNS www (
Domain = "veritas.com"
Alias = www
Hostname = north
TSIGKeyFile a= "/var/tsig/Kveritas.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, mountd, statd, and lockd daemons required by all exported NFS file systems. Always configure the NFSRestart agent along with NFS agent. The NFS resource should not be configured alone in a group.

Prerequisites for NFS lock recovery

If you plan on using lock recovery on a Linux system, store locking information on shared storage so that it is accessible to the system where NFS fails over. Using this information, NFS carries out lock recovery. See "Configuring NFS lock on shared storage" on page 81 for an NFS lock recovery example configuration.

Agent functions

- Online
  Starts nfsd and mountd daemons for all kernels. If daemons are not running, the agent starts the daemons and exits.
  - For Red Hat, the agent also starts the lockd and statd daemons.
  - For Suse, the agent also starts the lockd daemon. If the Address attribute is set, then sm-notify is started. The sm-notify daemon sends lock recovery notifications, which then perform their jobs and terminate automatically.
- Offline
  Stops the nfsd and mountd daemons for all kernels.
  - For Red Hat, the agent also stops the lockd and statd daemons.
  - For Suse, the agent also stops the lockd daemon.
- Monitor
  Monitors versions 2 and 3 of the nfsd daemon and versions 1, 2, and 3 of the mountd daemon for all kernels.
  - For Red Hat, the agent also monitors the lockd and statd daemons. Monitors versions 1, 3, 4 of the lockd daemon and version 1 of the statd daemon. When the value of the NFSv4Support attribute is 1, nfsd version 4 is also monitored.
  - For Suse, the agent also monitors the lockd daemon. Monitors versions 1, 3, 4 of the lockd daemon. Monitors version 1 of statd.

- Clean
  Stops nfsd and mountd daemons for all kernels.
  - For Red Hat, the agent also stops the lockd and statd daemons.
  - For Suse, the agent also stops the lockd daemon.

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.
- FAULTED
  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 attributes	Description
Address	Set the attribute to the name or the IP address that the clients use to mount file systems. Type and dimension: string-scalar Example: "11.124.205.20"
GracePeriod	Required when the value of the NFSRestart attribute is 1. GracePeriod specifies the amount of time that lock recovery is allowed by the NFS server after its reboot. Type and dimension: integer-scalar Default: 90

Table 4-1 Optional attributes

Optional attributes	Description
LockFileTimeout	The NFS and the NFSRestart agents require a synchronization mechanism when the group to which they belong is in transition, for example going online or coming offline. A file serves as this synchronization mechanism. The LockFileTimeout attribute specifies the maximum time that the synchronization file exists. Type and dimension: integer-scalar Default: 180
NFSSecurity	Specifies whether to start the NFS security daemon rpc.svcgssd or not. Type and dimension: boolean-scalar Default: 0
NFSv4Support	Specifies whether to start the NFSv4 daemon rpc.idmapd or not and whether to monitor nfsd version 4. Type and dimension: boolean-scalar Default: 0

Resource type definition

```
type NFS (
    static int RestartLimit = 1
    static str Operations = OnOnly
    static str ArgList[] = { Nproc, Address, GracePeriod,
    NFSSecurity, NFSv4Support, LockFileTimeout }
    int Nproc = 8
    str Address
    int GracePeriod = 90
    boolean NFSSecurity = 0
    boolean NFSv4Support = 0
    int LockFileTimeout = 180
)
```

Sample configurations

Setting Nproc configuration

Refer to the configuration for Share agent for more information.

```
NFS nfs_group_16 (
    Nproc = 16
)
```

Configuring NFS lock on shared storage

```
include "types.cf"

cluster vcs_cluster (
    CounterInterval = 5
    )

system sysa(
    )

system sysb(
    )

group test_grp (
    SystemList = { sysa=0, sysb=1 }
    )

    DiskGroup test_dg (
        DiskGroup = test_dg
        )

    IP test_ip (
        Device = eth0
        Address = "10.182.13.28"
        NetMask = "255.255.240.0"
        )

    Mount test_mnt (
        MountPoint = "/test_mnt"
        BlockDevice = "/dev/vx/dsk/test_dg/test_vol"
        FSType = ext3
        MountOpt = rw
        FsckOpt = "-y"
        )

    Mount test_lockinfo_mnt (
        MountPoint = "/lockinfo"
        BlockDevice = "/dev/vx/dsk/test_dg/test_lockinfo_vol"
        FSType = ext3
        MountOpt = rw
        FsckOpt = "-y"
```

```
    )
NFS test_nfs (
    Address = "10.182.13.28"
    )

NFSRestart test_nfsrestartres (
    NFSLockFailover  = 1
    LocksPathName = "/test_mnt"
    NFSRes = test_nfs
    )

Share test_share (
    PathName = "/test_mnt"
    Options = "-o rw"
    )

Volume test_lockinfo_vol (
    Volume = test_lockinfo_vol
    DiskGroup = test_dg
    )
Volume test_vol (
    Volume = test_vol
    DiskGroup = test_dg
    )

test_nfsrestartres requires test_ip
test_nfsrestartres requires test_lockinfo_mnt
test_lockinfo_mnt requires test_lockinfo_vol
test_lockinfo_vol requires test_dg
test_ip requires test_share
test_share requires test_nfs
test_share requires test_mnt
test_mnt requires test_vol
test_vol requires test_dg
```

NFSv4Support attribute configuration

This sample configuration is for NFS when the NFSv4Support attribute is 1. Note that the share test_share0 has fsid = 0 in Options attribute. This indicates that /home/export is root of all the exports. The client needs to mount only this root filesystem instead of mounting all shares individually.

The syntax is:

```
mount -t nfs4 <server>:/   <mountpoint>
```

Note that path after colon(:) is always /

Also note that all the filesystems other then the root filesystem should have the nohide option set in Options attribute of share resources. Set the nohide option so that authentic clients can seamlessly move through the tree of exported filesystems just by mounting the root filesystem.

```
include "types.cf"
```

```
cluster vcs_139 (
    UserNames = { admin = IJKcJEjGKfKKiSKeJH, a = gJJd }
    Administrators = { admin, a }
    )
system vcslinux139 (
    )
system vcslinux140 (
    )
group nfstest (
    SystemList = { vcslinux139 = 0, vcslinux140 = 1 }
    )
    DiskGroup test_dg1 (
        DiskGroup = nfsdg
        )
    IP test_ip (
        Device = eth0
        Address = "10.212.88.37"
        NetMask = "255.255.254.0"
        )
    Mount test_mnt0 (
        MountPoint = "/home/export"
        BlockDevice = "/dev/vx/dsk/nfsdg/vol0"
        FSType = vxfs
        MountOpt = rw
        FsckOpt = "-n"
        )
    Mount test_mnt1 (
        MountPoint = "/home/export/nshare1"
        BlockDevice = "/dev/vx/dsk/nfsdg/vol1"
        FSType = vxfs
        MountOpt = rw
        FsckOpt = "-n"
        )
    Mount test_mnt2 (
        MountPoint = "/home/export/nshare2"
        BlockDevice = "/dev/vx/dsk/nfsdg/vol2"
        FSType = vxfs
        MountOpt = rw
        FsckOpt = "-n"
        )
    Mount test_mnt3 (
        MountPoint = "/home/export/nshare3"
        BlockDevice = "/dev/vx/dsk/nfsdg/vol3"
        FSType = vxfs
        MountOpt = rw
        FsckOpt = "-n"
        )
    NFS test_nfs (
            Nproc = 10
        NFSv4Support = 1
        )
    NFSRestart test_nfsres (
```

```
    NFSRes = test_nfs
    )
NIC test_nic (
    Device = eth0
    )
Share test_share0 (
    PathName = "/home/export"
    Options = "rw,nohide,fsid=0"
    )
Share test_share1 (
    PathName = "/home/export/nshare1"
    Options = "rw,nohide"
    )
Share test_share2 (
    PathName = "/home/export/nshare2"
    Options = "rw,nohide"
    )
Share test_share3 (
    PathName = "/home/export/nshare3"
    Options = "rw,nohide"
    )
Volume test_vol0 (
    DiskGroup = nfsdg
    Volume = vol0
    )
Volume test_vol1 (
    DiskGroup = nfsdg
    Volume = vol1
    )
Volume test_vol2 (
    DiskGroup = nfsdg
    Volume = vol2
    )
Volume test_vol3 (
    DiskGroup = nfsdg
    Volume = vol3
    )
    test_ip requires test_nic
    test_ip requires test_share0
    test_ip requires test_share1
    test_ip requires test_share2
    test_ip requires test_share3
    test_mnt0 requires test_vol0
    test_mnt1 requires test_mnt0
    test_mnt1 requires test_vol1
    test_mnt2 requires test_mnt0
    test_mnt2 requires test_vol2
    test_mnt3 requires test_mnt0
    test_mnt3 requires test_vol3
    test_nfsres requires test_ip
    test_share0 requires test_mnt0
    test_share0 requires test_nfs
```

```
    test_share1 requires test_mnt1
    test_share1 requires test_nfs
    test_share2 requires test_mnt2
    test_share2 requires test_nfs
    test_share3 requires test_mnt3
    test_share3 requires test_nfs
    test_vol0 requires test_dg1
    test_vol1 requires test_dg1
    test_vol2 requires test_dg1
    test_vol3 requires test_dg1


// resource dependency tree
//
//      group nfstest
//      {
//      NFSRestart test_nfsres
//          {
//          IP test_ip
//              {
//              NIC test_nic
//              Share test_share0
//                  {
//                  Mount test_mnt0
//                       {
//                       Volume test_vol0
//                         {
//                         DiskGroup test_dg1
//                         }
//                       }
//                  NFS test_nfs
//                  }
//              Share test_share1
//                  {
//                  Mount test_mnt1
//                      {
//                      Mount test_mnt0
//                          {
//                          Volume test_vol0
//                              {
//                              DiskGroup test_dg1
//                              }
//                          }
//                      Volume test_vol1
//                          {
//                          DiskGroup test_dg1
//                          }
//                      }
//                  NFS test_nfs
//                  }
//              Share test_share2
//                  {
```

```
//                  Mount test_mnt2
//                      {
//                      Mount test_mnt0
//                          {
//                          Volume test_vol0
//                              {
//                              DiskGroup test_dg1
//                              }
//                          }
//                          Volume test_vol2
//                          {
//                          DiskGroup test_dg1
//                          }
//                      }
//                  NFS test_nfs
//                  }
//              Share test_share3
//                  {
//                  Mount test_mnt3
//                      {
//                      Mount test_mnt0
//                          {
//                          Volume test_vol0
//                              {
//                              DiskGroup test_dg1
//                              }
//                          }
//                          Volume test_vol3
//                              {
//                              DiskGroup test_dg1
//                              }
//                      }
//                  NFS test_nfs
//                  }
//              }
//          }
//      }
```

NFSRestart agent

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

If you have configured the NFSRestart agent for lock recovery, the NFSRestart agent starts the smsyncd daemon. The daemon copies the NFS locks from the shared-storage to the local directory (/var/lib/nfs) and vice-versa.

Dependencies

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, NFS, and Share agents must be in same service group.

Agent functions

- Online
  - Terminates statd and lockd.
  - If the value of the NFSLockFailover attribute is 1, it copies the locks from the shared storage to the /var/lib/nfs directory.
  - Starts the statd and lockd daemons.
  - Starts the smsyncd daemon to copy the contents of the /var/lib/nfs directory for Linux to the shared storage (LocksPathName) at regular, two-second intervals if the value of the NFSLockFailover attribute is 1.
- Monitor
  It monitors the smsyncd daemon if the value of the NFSLockFailover attribute is 1.
- 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 if the daemon is running.
- 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 if the daemon is running.
- nfs_postoffline
  - Restarts all the required NFS daemons if needed.

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.
- FAULTED
  Indicates that the NFS daemons are not running properly.

Attributes

Table 4-2 Optional attributes

Optional attribute	Description
LocksPathName	The path name of the directory to store the NFS lock information. This attribute is required when the value of the NFSLockFailover attribute is 1. The path that you specify for the LocksPathName attribute should be on shared storage. This is to ensure that it is accessible to all the systems where the NFSRestart resource fails over. Type and dimension: string-scalar
NFSLockFailover	NFS Lock recovery is done for all the Share resources that are configured in the group of this resource. Type and dimension: boolean-scalar Default: 0
NFSRes	Name of the NFS resource on the system. This attribute is required if the value of the NFSLockFailover attribute is 1. Type and dimension: string-scalar

Resource type definition

```
type NFSRestart (
```

```
    static str ArgList[] = { "NFSRes:Nproc", "NFSRes:Address",
    "NFSRes:GracePeriod", "NFSLockFailover", LocksPathName}
    str NFSRes
    str LocksPathName
    boolean NFSLockFailover = 0
)
```

Sample configurations

For NFS lock recovery:

```
NFSRestart nfsrestart (
    NFSRes = nfsres
    LocksPathName="/shared_mnt/lockinfo"
    NFSLockFailover = 1
)
```

For no NFS lock recovery:

```
NFSRestart nfsrestart (
    NFSRes = nfsres
)
```

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.

Dependencies

Share resources depend on NFS. In NFS service group, IP, IPMultiNIC, and IPMultiNICB resources depend on Share resources.

Agent functions

- Online
  Exports (shares) a directory to the specified client.
- Offline
  Unshares the exported directory from the client.
- Monitor
  Verifies that the shared directory is exported to the client.
- Clean
  Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

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.

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"

Table 4-4 Optional attributes

Optional attribute	Description
Client	The Share agent accepts as many clients as the user wishes provided all the clients are exported the same 'PathName'. Client or host where the directory specified by PathName is exported. The client can be a wild card (*) or a fully qualified domain name (FQDN) including the host name. Type and dimension: string-scalar Example: If "outland" is the host name, the FQDN hostname is outland.veritas.com.
Options	Options to the `exportfs` command. When specifying multiple options, separate them with commas, for example: "rw, no_root_squash" For more information about the `exportfs` command and its options, refer to the exportfs manpage. Type and dimension: string-vector Default = "ro, async, wdelay, root_squash"

Table 4-4 Optional attributes

Optional attribute	Description
OtherClients	The Client attribute can be assigned one FQDN host name, whereas multiple FQDN host names can be assigned to the 'OtherClients' field. A combination of 'Client' and 'OtherClients' can be used to specify the host names. If both of the Client and OtherClients attributes are left unspecified, the PathName is exported to the world (*). Type and dimension: string-vector

Resource type definition

```
type Share (
    static str ArgList[] = { PathName, Client, OtherClients,
    Options }
    str PathName
    str Client
    str OtherClients[]
    str Options
)
```

Sample configurations

Configuration 1

```
Share ShareMnt (
    PathName = "/mnt"
    Client = vcslinux2
    Options = rw
)
```

Configuration 2

In this example, a disk group and the volumes defined on it form the resource dg_dg1. Mount volumes defined on dg_dg1 at /extdisk1 and /extdisk2 and share them with clients for NFS mounting. This configuration ensures that the mounting process occurs prior to exporting, that NFS is online prior to sharing the directories, and that the IP address is online after sharing the directories.

```
system sysA

system sysB
```

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

    DiskGroup dg_dg1 (
        DiskGroup = dg1
        StartVolumes = 1
        StopVolumes = 1
        )

    IP ip_172_29_9_100 (
        Device = eth0
        Address = "172.29.9.100"
        )

    Mount mount_extdisk1 (
        MountPoint = "/extdisk1"
        BlockDevice = "/dev/vx/dsk/dg1/vol1"
        FSType = vxfs
        )

    Mount mount_extdisk2 (
        MountPoint = "/extdisk2"
        BlockDevice = "/dev/vx/dsk/dg1/vol2"
        FSType = vxfs
        )

    NFS nfs_groupA (
        Nproc = 16
        )

    NIC nic_groupA_eth0 (
        Device = eth0
        )

    Share share_extdisk1 (
        PathName = "/extdisk1"
        Client = "172.29.9.93"
        Options = "rw,no_root_squash"
        )

    Share share_extdisk2 (
        PathName = "/extdisk2"
        Client = "172.29.9.93"
        Options = "rw,no_root_squash"
        )

    ip_172_29_9_100 requires nic_groupA_eth0
    ip_172_29_9_100 requires share_extdisk1
    ip_172_29_9_100 requires share_extdisk2
```

```
mount_extdisk1 requires dg_dg1
mount_extdisk2 requires dg_dg1
share_extdisk1 requires mount_extdisk1
share_extdisk1 requires nfs_groupA
share_extdisk2 requires mount_extdisk2
share_extdisk2 requires nfs_groupA


// resource dependency tree
//
//  group groupA
//  {
//  IP ip_172_29_9_100
//        {
//        NIC nic_groupA_eth0
//        Share share_extdisk1
//                {
//                Mount mount_extdisk1
//                        {
//                        DiskGroup dg_dg1
//                        }
//                NFS nfs_groupA
//                }
//                Share share_extdisk2
//                {
//                Mount mount_extdisk2
//                        {
//                        DiskGroup dg_dg1
//                        }
//                NFS nfs_groupA
//                }
//        }
//  }
```

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 provides 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.

Platforms

Linux

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 start using 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.
- 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.

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

Starts, stops, and monitors the smbd process as a daemon. Only one resource of this type is permitted.

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.

Agent functions

- Online
  Starts the smbd daemon at specified 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/samba/smb.conf"
LockDir	Lock directory of Samba. Samba stores the files smbd.pid, nmbd.pid in this directory. Type and dimension: string-scalar Example: "/var/run"

Table 4-6 Optional attributes

Optional attribute	Description
IndepthMonitorCyclePeriod	Number of monitor cycles after which the in-depth monitoring is performed. For example, the value 5 indicates that the agent monitors the resource in-depth every five monitor cycles. The value 0 indicates that the agent will not perform in-depth monitoring for the resource. Type and dimension: integer-scalar Default: 5
Ports	Ports where Samba accepts connections. To run Samba over NBT (NetBios over TCP/IP), set this attribute to 139. To run Samba directly over TCP/IP, set this attribute to 445. For Samba version less than 3.0, exactly one value must be provided. Type and dimension: integer-vector Default: 139, 445

Table 4-6 Optional attributes

Optional attribute	Description
ResponseTimeout	Number of seconds the agent waits to receive the session response packet after sending the session request packet. For example, the value 5 indicates that the agent waits for five seconds before receiving the session response packet. Configure this attribute if in-depth monitoring is enabled. Type and dimension: integer-scalar Default: 10

Resource type definition

```
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 configuration

```
SambaServer samba_server (
    Ports = { 139 }
    ConFile = "/etc/samba/smb.conf"
    LockDir = "/var/run"
    IndepthMonitorCyclePeriod = 3
    ResponseTimeout = 15
)
```

SambaShare agent

Adds, removes, and monitors a share by modifying the specified Samba configuration file.

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.

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.
- FAULTED
  Indicates and invalid Samba configuration file.
- UNKNOWN
  Indicates that the agent could not determine the state of the resource.

Attributes

Table 4-7 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

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

Sample configuration

```
SambaShare samba_drive (
    SambaServerResName = samba_server
    ShareName = share1
    ShareOptions = "path=/shared; public=yes; writable = yes"
    )
```

NetBIOS agent

Starts, stops, and monitors the nmbd daemon. Only one resource of this type is permitted.

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 IP and IPMultiNIC resources.

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

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 4-8 Required attributes

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

Table 4-9 Optional attributes

Optional attribute	Description
DomainMaster	If set to 1, this flag causes the agent to set Samba as Domain Master. Type and dimension: integer-scalar Default: 0
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

Resource type definition

```
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_server
    NetBiosName = sambaX
    NetBiosAliases = { samba1, samba2 }
    Interfaces = { 172.16.7.53/24, 172.16.7.54/255.255.255.0 }
    WinsSupport = 0
    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

Brings an Apache Server online and offline, and monitors the processes. The Apache Web server agent consists of resource type declarations and agent scripts.

This agent supports the Apache HTTP server 1.3, 2.0, and 2.2. It also supports the IBM HTTP Server 1.3 and 2.0.

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

Before you use this agent:

- Install the Apache server on shared disk.
- Verify that the floating IP has the same subnet as that of 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.
- Install the ACC Library 4.1.04.0 (VRTSacclib) if it is not already installed. If the ACC Library needs to be installed or updated, the library and its documentation can be obtained from the agent software media.
- Remove prior 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.

Dependency

This type of resource depends on IP and Mount resources.

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 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.

State definitions

- ONLINE

  Indicates that the Apache server is running.

- OFFLINE

  Indicates that the Apache server is not running.

- 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"
HostName	Virtual host name that is assigned to the Apache server instance. The host name is used in second-level monitoring to establish a socket connection with the Apache HTTP server. Specify this attribute only if the SecondLevelMonitor is set to 1 (true). Type and dimension: string-scalar Example: "web1.veritas.com"
Port	Port number where the Apache HTTP server instance listens. The port number is used in second-level monitoring to establish a socket connection with the server. Specify this attribute only if SecondLevelMonitor is set to 1 (true). Type and dimension: integer-scalar Default: 80 Example: "80"

Table 5-1 Required attributes

Required attribute	Description
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"
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"

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 }

Table 5-2 Optional attributes

Optional attribute	Description
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 -k start -DSSL` Set to 0 (false) it excludes the `-DSSL` option from the command. Type and dimension: boolean-scalar Default: 0 Example: "1"
EnvFile	Full path and file name of the file that is sourced prior to 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 prior to executing httpd. Specifying this attribute is optional. If EnvFile is specified, the login shell for user root must be Bourne, Korn, or C shell. Type and dimension: string-scalar Example: "/apache/server1/bin/envvars"
SecondLevelMonitor	Enables second-level monitoring for the resource. Second-level monitoring is a deeper, more thorough state check of the Apache HTTP server performed by issuing an HTTP GET request on the web server's root directory. Valid attribute values are **1** (true) and **0** (false). Specifying this attribute is required. Type and dimension: boolean-scalar Default: 0 Example: "1"
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 specified, 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"

Table 5-2 Optional attributes

Optional attribute	Description
SecondLevelTimeout	Number of seconds monitor entry point will wait on the execution of second-level monitor. If the second-level monitor program does not return to the calling monitor entry point before the SecondLevelTimeout window expires, the monitor entry point will no longer block on the program sub-process but will report that the resource is offline. The value should be sufficiently high to allow second level monitor enough time to complete, but the value should also be less than the value specified by the agent's MonitorTimeout. Type and dimension: integer-scalar Default: 30

Resource type definition

```
type Apache (
    static str ArgList[] = { ResLogLevel, State, IState, httpdDir,
    SharedObjDir, EnvFile, HostName, Port, User,
    SecondLevelMonitor, SecondLevelTimeout, ConfigFile, EnableSSL,
    DirectiveAfter, DirectiveBefore}
    str ResLogLevel = "INFO"
    str httpdDir
    str SharedObjDir
    str EnvFile
    str HostName
    int Port = 80
    str User
    boolean SecondLevelMonitor
    int SecondLevelTimeout = 30
    str ConfigFile
    boolean EnableSSL
    str DirectiveAfter{}
    str DirectiveBefore{}
)
```

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 by searching for the existence of the parent httpd daemon and 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 agent attributes Host and Port. Once 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 the ACC Library

The agent functions for the Apache HTTP server depend on a set of Perl modules known as the ACC Library. The ACC Library contains common, reusable functions that perform tasks such as process identification, logging, and system calls.

When you install the ACC library in a VCS environment, you must install the ACC library package before you install the agent.

To install or update the ACC library package, locate the library and related documentation on the agent disc and in the compressed agent tar file.

Sample configurations

Running two versions of httpd

This example shows how two versions of `httpd` can run from different locations. In group Apache_1, httpd runs from Port 80, the default location. The configuration file in /usr/local/apache/conf/httpd.conf should indicate DocumentRoot, address, port, and other parameters. In group Apache_2, httpd runs from /home/web/apache. The PID file for this is created in /home/web/apache/logs/httpd.pid. The configuration file in /home/web/apache/conf/httpd.conf should define parameters for running this version of httpd.

Each Apache resource requires an online IP resource. In this example, each Apache resource requires an online mount resource to mount block devices from disks reserved by the Disk Reservation agent.

```
system sysa

system sysb


group Apache_1 (
    SystemList = { sysa ,sysb}
    AutoStartList = { sysa}
)

Apache myapacheWeb
    ServerRoot = "/usr/local/apache"
    DetailMonitor = 10
```

```
    PidFile = "logs/httpd.pid"
    ConfigFile = "conf/httpd.conf"
    Address = "192.168.50.50"

)

IP myapacheIP(
    Device = "eth0"
    Address="192.168.50.50"
    NetMask="255.255.255.0"

)

NIC myapacheNIC(
    Device="eth0"
    NetworkHosts={"172.29.9.178","172.29.9.179"}
)

Mount myapacheMnt(
    MountPoint="/mnt/apache/"
    BlockDevice="/dev/sdd2"

)

DiskReservation myapacheDiskRes(
    Disks ="/dev/sdd"

)
myapacheMnt requires myapacheDiskRes
myapacheIP requires myapacheNIC
myapacheWeb requires myapacheIP
myapacheWeb requires myapacheMnt

group Apache_2 (
    SystemList = { sysa,sysb}
    AutoStartList = { sysa}
    )

Apache myapacheWeb2(
    ServerRoot = "/home/web/apache"
    DetailMonitor = 10
    PidFile = "logs/httpd.pid"
    ConfigFile = "conf/httpd.conf"
    Address = "192.168.60.50"

)

IP myapacheIP2(
    Device = "eth1"
    Address="192.168.60.50"
    NetMask="255.255.255.0"
```

```
)

NIC myapacheNIC2(
    Device="eth1"
)

Mount myapacheMnt2(
    MountPoint="/mnt/apache1/"
    BlockDevice="/dev/sdc3"

)

DiskReservation myapacheDiskRes2(
    Disks ="/dev/sdc"

)
myapacheMnt2 requires myapacheDiskRes2
myapacheIP2 requires myapacheNIC2
myapacheWeb2 requires myapacheIP2
myapacheWeb2 requires myapacheMnt2
```

Sample main.cf file

```
include "types.cf"
cluster Cluster1 (
    UserNames = { admin = xxxxxx }
    Administrators = { admin }
    CounterInterval = 5
    )
system SystemA (
    )
system SystemB (
    )
group Web1 (
    SystemList = { SystemA = 0, SystemB = 1 }
    )
    DiskGroup Web1_dg (
        DiskGroup = web1
        )
    IP Web1_ip (
        Device = hme0
        Address = "1.1.1.1"
        NetMask = "255.255.255.0"
        )
    Mount Web1_mnt (
        MountPoint = "/apache/srvr01"
        BlockDevice = "/dev/vx/dsk/web1/web1"
        FSType = vxfs
        FsckOpt = "-y"
        )
    NIC Web1_nic (
        Device = hme0
```

```
    NetworkType = ether
    )
VProApache Web1_http (
    HostName = web1
    Port = 80
    SecondLevelMonitor = 1
    SecondLevelTimeout = 25
    httpdDir = "/apache/srvr01/bin"
    EnvFile = "/apache/srvr01/bin/envvars"
    ConfigFile = "/apache/srvr01/conf/httpd.conf"
    )
Web1_ip   requires Web1_nic
Web1_mnt  requires Web1_dg
Web1_http requires Web1_ip
Web1_http requires Web1_mnt
```

Application agent

Brings applications online, takes them offline, and monitors their status. Enables you to specify different executables for the online, offline, and monitor routines, because most applications have executables to start and stop the application. The executables must exist locally on each node.

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

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

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

Virtual fire drill

The virtual 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 Application resources, the virtual 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 virtual fire drill see the *VCS User's Guide*.

Dependencies

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

Agent functions

- Online
  Runs the StartProgram with the specified parameters in the context of the specified user.

- Offline
  Runs the StopProgram with the specified parameters in the context of the specified user.
- Monitor
  If you specify the MonitorProgram, the agent executes the user-defined MonitorProgram in the user-specified context. If you specify PidFiles, the routine verifies that the process ID found in each listed file is running. If you specify MonitorProcesses, the routine verifies that each listed process is running in the context you specify.
  Use any one, two, or three of these attributes to monitor the application.
  If any one process 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 (specified in MonitorProcesses) running with the user ID specified in the User attribute are killed. If the CleanProgram is defined, the agent executes the CleanProgram.

State definitions

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

Attributes

Table 5-3 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 are separated by spaces. Type and dimension: string-scalar Example: "/usr/sbin/samba start"
StopProgram	The executable, created locally on each node, that stops the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and are separated by spaces. 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 120.

Table 5-4 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 are separated by spaces. Type and dimension: string-scalar

Table 5-4 Optional attributes

Optional attribute	Description
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 name displayed by the `ps -ef` command for the process. Type and dimension: string-vector Example: "nmbd"
MonitorProgram	The executable, created locally on each node, which monitors the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and are separated by spaces. MonitorProgram can return the following VCSAgResState values: OFFLINE value is 100; ONLINE values range from 101 to 110 (depending on the confidence level); 110 equals confidence level of 100%. Any other value = UNKNOWN. Type and dimension: string-scalar
PidFiles	A list of PID files that contain the process ID (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 script may return an incorrect result. If this occurs, increase the ToleranceLimit in the resource definition. Type and dimension: string-vector

Table 5-4 Optional attributes

Optional attribute	Description
User	The user ID for running StartProgram, StopProgram, MonitorProgram, and CleanProgram. The processes specified in the MonitorProcesses list must run in the context of the specified user. Monitor checks the processes to make sure they run in this context. Type and dimension: string-scalar Default: root

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
    str StartProgram
    str StopProgram
    str CleanProgram
    str MonitorProgram
    str PidFiles[]
    str MonitorProcesses[]
)
```

Sample configurations

Configuration 1

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

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

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. The agent also monitors the smbd and nmbd processes.

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

Process agent

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

Virtual fire drill

The virtual 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 virtual 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 virtual fire drill see the *VCS User's Guide*.

Dependencies

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

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.

- 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 5-5 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 256 characters. Type and dimension: string-scalar Example: "/usr/sbin/proc1"

Table 5-6 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. Type and dimension: string-scalar

Table 5-6 Optional attributes

Optional attribute	Description
PidFile	The file that contains the process ID for the monitoring process. Specify the PidFile attribute for the monitoring process to use the Pid. Otherwise, to complete the monitoring process the agent uses the ps output. Note that when you use scripts, or other indirect mechanisms, to start processes, you must set the PidFile attribute if the ps output is different from the configured values for the PathName or Arguments attributes. Type and dimension: string-scalar Example: "/var/lock/sendmail.pid"
Priority	Priority with which the process will run. Priority values range between -20 (highest) to +19 (lowest). Type and dimension: string-scalar Default: 10
UserName	Owner of the process. The process runs with the user ID. Type and dimension: string-scalar Default: root

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 = 10
    str PidFile
)
```

Sample configurations

Configuration

In this example, the Process agent starts, stops, and monitors sendmail. This process is started with two arguments as determined in the Arguments attribute. The pid stored in the PidFile attribute is used to monitor the sendmail process.

```
Process sendmail (
    PathName = "/usr/sbin/sendmail"
    Arguments = "-bd -q30m"
    PidFile = "/var/run/sendmail.pid"
)
```

ProcessOnOnly agent

Starts and monitors a user-specified process.

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.
- FAULTED
  Indicates that the process has unexpectedly terminated.
- UNKNOWN
  Indicates that the agent can not determine the state of the process.

Attributes

Table 5-7 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. The PathName attribute must not exceed 256 characters. Type and dimension: string-scalar

Table 5-8 Optional attributes

Optional attribute	Description
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. Type and dimension: string-scalar Example: "-bd -q30m"
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
PidFile	The file that contains the process ID for the monitoring process. Specify the PidFile attribute for the monitoring process to use the Pid. Otherwise, to complete the monitoring process the agent uses the ps output. Note that when you use scripts, or other indirect mechanisms, to start processes, you must set the PidFile attribute when the ps output is different from the configured values for the PathName or Arguments attributes. Type and dimension: string-scalar Example: "/var/lock/sendmail.pid"
Priority	Priority with which the process will run. Priority values range between -20 (highest) to +19 (lowest). Type and dimension: string-scalar Default: 10

Table 5-8 Optional attributes

Optional attribute	Description
UserName	Owner of the process. The process runs with the user ID. Type and dimension: string-scalar Default: root

Resource type definition

```
type ProcessOnOnly (
    static str ArgList[] = { PathName, Arguments, UserName,
    Priority, PidFile, IgnoreArgs }
    static str Operations = OnOnly
    str PathName
    str Arguments
    str UserName = root
    str Priority = 10
    str PidFile
    boolean IgnoreArgs = 0
)
```

Sample configurations

Configuration 1

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

Configuration 2

```
include "types.cf"

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

    ProcessOnOnly Process1 (
        PathName = "/usr/local/bin/myprog"
```

```
    Arguments = "arg1 arg2"
    )

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

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

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.

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

Dependency

The NotifierMngr resource depends 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 name of the SNMP manager and the severity level of the messages to be delivered to the SNMP manager. The severity levels of messages are `Information`, `Warning`, `Error`, and `SevereError`. Specifying a given severity level for messages generates delivery of all messages of equal or higher severity. 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. SmtpServer is a required attribute if SnmpConsoles is not specified; otherwise, SmtpServer is an optional attribute. You can specify both SmtpServer and SnmpConsoles if desired. 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 numbers. 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@veritas.com" = SevereError, "admin@veritas.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 you need to set the SmtpVrfyOff to `1` in order for the SmtpReturnPath value to take effect. Type and dimension: string-scalar Example: "usera@example.com"
SmtpServerTimeout	This attribute represents the time in seconds notifier waits for a response from the mail server for the SMTP commands it has sent to the mail server. This value can be increased if you notice that the mail server is taking a longer duration to reply back to the SMTP commands sent by notifier. Type and dimension: integer-scalar Default: 10
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 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 `SnmpConsole: snmpserv`. In this example, only messages of SevereError level are sent to the SmptServer (`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_eth0 (
        Enabled = 1
        Device  = eth0
        )

group Grp1 (
    SystemList = { north, south }
    AutoStartList = { north }
    )

    Proxy nicproxy(
    TargetResName = "NicGrp_eth0"
```

```
)

NotifierMngr ntfr (
    SnmpConsoles = { snmpserv = Information }
    SmtpServer = "smtp.example.com"
    SmtpRecipients = { "vcsadmin@example.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.

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 the 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 it is loaded into the Web server. This parameter is returned as the exit value of the online script, which inform VCS of the time it needs to wait before calling monitor on the Web application resource. This attribute value is typically at least five seconds. Type and dimension: integer-scalar

Resource type definition

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

Sample configuration

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

Proxy agent

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

A Proxy resource can only point to None or OnOnly type of resources, and can reside in a failover/parallel group.

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 int OfflineMonitorInterval = 60
    static str ArgList[] = { TargetResName, TargetSysName,
    "TargetResName:Probed", "TargetResName:State" }
    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 3

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 @vcslx3 = { eth0 = "192.123.8.42", eth3 =
        "192.123.8.42" }
        Device @vcslx4 = { eth0 = "192.123.8.43", eth3 =
        "192.123.8.43" }
        NetMask = "255.255.248.0"
    NetworkHosts = { "192.123.10.129", "192.123.10.130" }
    )
```

```
    IPMultiNIC ip1 (
        Address = "192.123.10.177"
        MultiNICAResName = mnic
        NetMask = "255.255.248.0"
        )

    ip1 requires mnic


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

        IPMultiNIC ip2 (
            Address = "192.123.10.178"
            NetMask = "255.255.255.0"
            MultiNICAResName = mnic
            )
        Proxy proxy (
            TargetResName = mnic
            )
    ip2 requires proxy
```

Phantom agent

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

Agent functions

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

Resource type definition

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

Sample configurations

Configuration 1

```
Phantom (
)
```

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.

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 are:

- 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.
- Global groups are not supported as remote service groups.

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 152 for more information.
- Offline
  Takes the remote service group offline.
  See the “ControlMode” on page 152 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 151.
- Clean
  If the RemoteGroup resource faults, the Clean function takes the remote service group offline.
  See the “ControlMode” on page 152 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 6-6 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 Example: "www.example.com" or "11.183.12.214"
Port	The port on which 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
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 6-6 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 6-6 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 6-7 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 the user needs the RemoteGroup agent to communicate to a specific authentication broker, then set this attribute. Enter the information for the specific authentication broker in the format "IP:Port". Type: string-scalar Example: "128.11.295.51:1400"
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 6-8 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 *VCS 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 program support agents

Use the program support agents to provide high availability for program support resources.

ElifNone agent

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

Agent function

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

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—check's for the file's existence.

Agent functions

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

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

Creates, removes, and monitors files.

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.

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

Creates and monitors files.

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.
- Clean
  Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

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

I

K

L

M

N

O

P

R

S

T

V