Veritas™ Cluster Server Bundled Agents Reference Guide

HP-UX

5.0

N18453G

Veritas Cluster 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

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 1 Introduction

Resources and their attributes 13
Modifying agents and their resources 14
Attributes 14

Chapter 2 Storage agents

About the storage agents 17
DiskGroup agent 18
    Virtual fire drill 18
    Agent functions 18
    State definitions 19
    Attributes 20
    Resource type definition 21
    Sample configurations 21
        DiskGroup resource configuration 21
        DiskGroup, Volume, and Mount dependencies configuration 22
Volume agent 23
    Dependency 23
    Agent functions 23
    State definitions 23
    Attributes 24
    Resource type definition 24
    Sample configurations 24
        Configuration 24
LVMLogicalVolume agent 25
    Dependency 25
    Agent functions 25
    State definitions 25
    Attributes 26
    Resource type definition 26
    Physical volumes associated with volume groups 26
    Sample configurations 27
        Configuration 27

LVMVolumeGroup agent 28
Dependency 28
Agent functions 28
State definitions 28
Attributes 29
Resource type definition 29
Sample configurations 29
Configuration 1 29
Configuration 2: LVMVolumeGroup, LVMLogicalVolume, and Mount Dependencies 29
LVMCombo agent 31
Agent functions 31
State definitions 32
Attributes 32
Resource type definition 33
Physical volumes associated with volume groups 33
Sample configurations 33
Sample 1 33
Sample 2: LVMCombo and Mount Dependencies 33
Mount agent 35
Virtual fire drill 35
Agent functions 35
State definitions 36
Attributes 37
Resource type definition 39
Sample configurations 39
Configuration 39

Chapter 3 Network agents

About the network agents 41
Agent comparisons 41
IP and NIC agents 41
IPMultiNIC and MultiNICA agents 41
IPMultiNICB and MultiNICB agents 42
802.1Q trunking 43
IP agent 44
Virtual fire drill 44
Dependency 44
Agent functions 44
State definitions 45
Attributes 45
Resource type definition 47
Sample configurations 47

Configuration 1 47
NetMask in decimal (base 10) 47
NetMask in hexadecimal (base 16) 47
NIC agent 48
Virtual fire drill 48
Agent functions 48
State definitions 48
Attributes 49
Resource type definition 50
Sample configurations 50
Network Hosts 50
IPMultiNIC agent 51
Dependency 51
Agent functions 51
State definitions 51
Attributes 52
Resource type definition 53
Sample configuration: IPMultiNIC and MultiNICA 53
MultiNICA agent 54
Agent function 54
State definitions 54
Attributes 55
Resource type definition 57
MultiNICA notes 57
Using RouteOptions 58
Sample configurations 59
MultiNICA and IPMultiNIC 59
About the IPMultiNICB and MultiNICB agents 61
Checklist to ensure the proper operation of MultiNICB 61
IPMultiNICB agent 62
Dependencies 62
Requirements for IPMultiNICB 62
Agent functions 62
State definitions 63
Attributes 64
Resource type definition 65
Manually migrating a logical IP address 65
Sample configurations 65
Other sample configurations for IPMultiNICB and MultiNICB 65
MultiNICB agent 66
Agent functions 67
State definitions 67
Attributes 68

Resource type definition 70
Trigger script 71
IPMultiNICB and MultiNICB configuration 71
DNS agent 73
Agent functions 73
State definitions 73
Attributes 74
Resource type definition 75
Online query 75
Monitor scenarios 76
Sample web server configuration 76
Sample DNS configuration 76
Secure DNS update 77
Setting up secure updates using TSIG keys 77

Chapter 4 File share agents

About the file service agents 79
NFS agent 80
Agent functions 80
State definitions 80
Attributes 80
Resource type definition 81
Sample configurations 81
Configuration 81
NFSRestart agent 82
Dependencies 82
Agent functions 82
State definitions 83
Attributes 83
NFSRestart notes 83
Resource type definition 84
Sample configurations 84
Share agent 87
Dependencies 87
Agent functions 87
State definitions 87
Attributes 88
Resource type definition 88
Sample configurations 88
Configuration 88

Chapter 5 Service and application agents

About the service and application agents 89
Apache Web server agent 90
  Dependency 90
  Agent functions 91
  State definitions 91
  Attributes 92
  Resource type definition 95
  Detecting Application Failure 95
  About the ACC Library 96
  Sample configurations 96
Application agent 98
  Virtual fire drill 98
  Dependencies 98
  Agent functions 98
  State definitions 99
  Attributes 100
  Resource type definition 102
  Sample configurations 102
    Sample Configuration 1 102
    Sample Configuration 2 103
Process agent 104
  Virtual fire drill 104
  Dependencies 104
  Agent functions 104
  State definitions 104
  Attributes 105
  Resource type definition 106
  Sample configurations 106
    Configuration 1 106
    Sample configuration 2 107
ProcessOnOnly agent 108
  Agent functions 108
  State definitions 108
  Attributes 109
  Resource type definition 110
  Sample configurations 110
    Configuration 1 110

Chapter 6 Infrastructure and support agents

About the infrastructure and support agents 111
NotifierMngr agent 112

Dependency 112
Agent functions 112
State definitions 112
Attributes 113
Resource type definition 116
Sample configuration 117
Configuration 117
VRTSWebApp agent 119
Agent functions 119
State definitions 119
Attributes 120
Resource type definition 120
Sample configuration 120
Proxy agent 122
Agent functions 122
Attributes 122
Resource type definition 123
Sample configurations 123
Configuration 1 123
Configuration 2 123
Configuration 123
Phantom agent 125
Agent functions 125
Attribute 125
Resource type definition 125
Sample configurations 125
Configuration 1 125
Configuration 2 125
RemoteGroup agent 127
Dependency 127
Agent functions 128
State definitions 128
Attributes 129
Resource type definition 133

Chapter 7 Testing agents

About the program support agents 135
ElifNone agent 136
Agent function 136
Attributes 136
Resource type definition 136
Sample configuration 136
FileNone agent 137

Agent functions 137
Attribute 137
Resource type definition 137
Sample configuration 137
FileOnOff agent 138
Agent functions 138
Attribute 138
Resource type definition 138
Sample configuration 139
FileOnOnly agent 140
Agent functions 140
Attribute 140
Resource type definition 140
Sample configuration 140

Glossary 141

Index 143

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:

- "DiskGroup agent" on page 18
- "Volume agent" on page 23
- "LVMLogicalVolume agent" on page 25
- "LVMVolumeGroup agent" on page 28
- "LVMCombo agent" on page 31
- "Mount agent" on page 35

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*.

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 auto:cdsdisk
  FreeSize 12765712
  ```

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"

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
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

Table 2-2 Optional attributes

Optional attributes	Description
StartVolumes	If value is 1, the DiskGroup online script starts all volumes belonging to that disk group after importing the group. Type and dimension: string-scalar Default: 1
StopVolumes	If value is 1, the DiskGroup offline script stops all volumes belonging to that disk group before deporting the group. Type and dimension: string-scalar Default: 1
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, tempUseFence,
    PanicSystemOnDGLoss }
    str DiskGroup
    str StartVolumes = 1
    str StopVolumes = 1
    boolean MonitorReservation = 0
    temp str tempUseFence = INVALID
    boolean PanicSystemOnDGLoss = 1
)
```

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"
)
```

DiskGroup, Volume, and Mount dependencies configuration

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

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

        Volume vres (
            Volume = vol1
            DiskGroup = dg2
            )

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

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

    mres requires vres
    vres requires dres
```

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 2-3 Required attributes

Required attribute	Description
DiskGroup	Name of the disk group that contains the volume. Type and dimension: string-scalar Example: `"sharedg"`
Volume	Name of the volume. Type and dimension: string-scalar Example: `"vol3"`

Resource type definition

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

Sample configurations

Configuration

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

LVMLogicalVolume agent

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

Dependency

LVMLogicalVolume resources depend on LVMVolumeGroup resources.

Agent functions

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

State definitions

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

Attributes

Table 2-4 Required attributes

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

Table 2-5 Optional attributes

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

Resource type definition

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

Physical volumes associated with volume groups

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

Use the following command to change the timeout:

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

For example:

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

Sample configurations

Configuration

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

LVMVolumeGroup agent

Activates, deactivates, and monitors LVM volume groups.

Dependency

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

Agent functions

- Online
  Activates a volume group. While each system in the cluster must import the volume group, each system does not need to activate it.
  This agent does not import volume groups because of the way LVM stores configuration information. Use the HP-UX SAM tool to import a volume group.
- Offline
  Deactivates a volume group with the `vgchange` command.
- Monitor
  Determines whether the volume group is available.
- Clean
  Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.

State definitions

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

Attributes

Table 2-6 Required attributes

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

Resource type definition

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

Sample configurations

Configuration 1

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

Configuration 2: LVMVolumeGroup, LVMLogicalVolume, and Mount Dependencies

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

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

        LVMLogicalVolume lvolres (
            LogicalVolume = lvol2
            VolumeGroup = vg01
            )

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

mres requires lvolres
lvolres requires lvgres
```

LVMCombo agent

Controls the activation and deactivation of the logical volumes and the Logical Volume group.

Agent functions

- Online

  Activates the volume group and any of the logical volumes that are not available. While each system in the cluster must import the volume group, each system should not activate it.

  This agent does not import volume groups because of the way LVM stores configuration information. Use the HP-UX SAM tool to import a volume group.
- Offline

  Deactivates the volume group, but does not deactivate the logical volumes. The logical volumes are automatically deactivated when the volume group is deactivated.
- Monitor

  If the volume group and all of the logical volumes are activated, the resource is online. Otherwise, the resource is reported offline.

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

State definitions

- ONLINE

  Indicates that the Volume Group and Logical Volumes are active.
- OFFLINE

  Indicates that the Volume Group and Logical Volumes are not active.
- UNKNOWN

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

Attributes

Table 2-7 Required attributes

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

Table 2-8 Optional attributes

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

Resource type definition

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

Physical volumes associated with volume groups

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

Use the following command to change the timeout:

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

For example:

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

Sample configurations

Sample 1

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

Sample 2: LVMCombo and Mount Dependencies

This sample configuration shows the LVMCombo and Mount dependencies:

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

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

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

mres requires lvmcmbres
```

Mount agent

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:
  ```
  bdf mount_point
  ```
  The output displays Mount resource information:
  ```
  Size Used Avail Use%
  ```
  To initiate the info agent function, set the InfoInterval timing to a value greater than 0. In this example, the info agent function executes every 60 seconds:
  ```
  haconf -makerw
  hatype -modify Mount InfoInterval 60
  ```

The command to retrieve information about the Mount resource is:

```
hares -value mountres ResourceInfo
```

Output includes:

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

State definitions

- ONLINE

  For the file system, indicates that the block device is mounted on the specified mount point.

  For the NFS client, indicates that the NFS remote client is mounted in the specified mount directory.

- OFFLINE

  For the file system, indicates that the block device is not mounted on the specified mount point.

  For the NFS client, indicates that the NFS remote client is not mounted in the specified mount directory.

- FAULTED

  For the file system, indicates that the block device has unexpectedly unmounted.

  For the NFS client, indicates that the NFS remote client has unexpectedly unmounted.

- UNKNOWN

  Indicates that a problem exists either with the configuration or the 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/vx/dsk/campus-dg1/campus-vol1" "/dev/vg02/lvol1"
FsckOpt	Required when the value of the FSType attribute is not nfs. For NFS file system, this attribute is optional. Options for `fsck` command. You must include `-y` or `-n` must as arguments to `fsck` for the resource to come online. The `-y` argument enables the VxFS file systems to perform a log replay before a full `fsck` operation. Refer to the manual page on the `fsck` command for more information. Type and dimension: string-scalar
FSType	Type of file system. Supports vxfs, hfs, or nfs. Type and dimension: string-scalar Example: "nfs"
MountPoint	Directory for mount point. Type and dimension: string-scalar Example: "/campus1"

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: integer-scalar Default: 1
MountOpt	Options for the `mount` command. Refer to the `mount` manual page for more information. Type and dimension: string-scalar Example: "rw"
SecondLevelMonitor	This attribute is only applicable to NFS client mounts. If set to 1, this attribute enables detailed monitoring of a NFS mounted file system. Type and dimension: boolean-scalar Default: 0
SecondLevelTimeout	This attribute is only applicable to NFS client mounts. This is the timeout (in seconds) for the SecondLevelMonitor attribute. 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: integer-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
    int SnapUmount
    int CkptUmount = 1
    boolean SecondLevelMonitor = 0
    int SecondLevelTimeout = 30
)
```

Sample configurations

Configuration

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

Network agents

This chapter contains the following:

- "About the network agents" on page 41
- "IP agent" on page 44
- "NIC agent" on page 48
- "IPMultiNIC agent" on page 51
- "MultiNICA agent" on page 54
- "About the IPMultiNICB and MultiNICB agents" on page 61
- "IPMultiNICB agent" on page 62
- "MultiNICB agent" on page 66
- "DNS agent" on page 73

About the network agents

Use network agents to provide high availability for networking resources.

Agent comparisons

IP and NIC agents

The IP and NIC agents:

- Monitor a single NIC

IPMultiNIC and MultiNICA agents

The IPMultiNIC and MultiNICA agents:

- Monitor single or multiple NICs

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

IPMultiNICB and MultiNICB agents

The IPMultiNICB and MultiNICB agents:

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

802.1Q trunking

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

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

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

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

IP agent

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.

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

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.
- 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 attributes	Description
Address	A virtual IP address, which is different from the base IP address, and which is associated with the interface. Type and dimension: string-scalar Example: "192.203.47.61"
Device	The name of the NIC device associated with the IP address. Contains the device name without an alias. Type and dimension: string-scalar Example: "lan0"

Table 3-2 Optional attributes

Required attributes	Description
ArpDelay	The number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about this IP address. Type and dimension: integer-scalar Default: 1
IfconfigTwice	Causes an IP address to be configured twice using an `ifconfig` `up-down-up` command sequence. Increases the probability of gratuitous ARP requests (generated by `ifconfig up`) to reach clients. Type and dimension: integer-scalar
NetMask	The netmask associated with the IP address of the resource. Specify the value of the netmask in decimal (base 10) or hexadecimal (base 16). **Note:** Symantec recommends that you specify a netmask for each virtual interface. Type and dimension: string-scalar Example: "255.255.210.0"
Options	Options for the `ifconfig` command. Type and dimension: string-scalar Example: "broadcast 192.203.15.255"

Resource type definition

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

Sample configurations

Configuration 1

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

NetMask in decimal (base 10)

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

NetMask in hexadecimal (base 16)

```
IP          IP_192_203_47_61 (
    Device = lan0
    Address = "192.203.47.61"
    NetMask = "0xfffff800"
    )
```

NIC agent

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

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*.

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

Before using this agent:

- Verify that the NIC has the correct administrative IP address and subnet mask.
- Verify that the NIC does not have built-in failover support. If it does, disable it.

Agent functions

- Monitor
  Tests the network card and network link. Pings the network hosts or broadcast address of the interface to generate traffic on the network. Counts the number of packets passing through the device before and after the address is pinged. If the count decreases or remains the same, the resource is marked FAULTED.

State definitions

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

Attributes

Table 3-3 Required attributes

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

Table 3-4 Optional attributes

Optional attribute	Description
NetworkHosts	List of hosts on the network that are pinged to determine if the network connection is alive. Enter the IP address of the host, instead of the host name, to prevent the monitor from timing out. DNS causes the ping to hang. If more than one network host is listed, the monitor returns ONLINE if at least one of the hosts is alive. Type and dimension: string-vector Example: "166.96.15.22" , "166.97.1.2"
NetworkType	Type of network. VCS currently only supports Ethernet. Type and dimension: string-scalar Default: "ether"
PingOptimize	Number of monitor cycles to detect if configured interface is inactive. Use PingOptimize when you have not specified network hosts. A value of 1 optimizes broadcast pings and requires two monitor cycles. A value of 0 performs a broadcast ping during each monitor cycle and detects the inactive interface within the cycle. Type and dimension: integer-scalar Default: 1

Resource type definition

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

Sample configurations

Network Hosts

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

IPMultiNIC agent

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.
- Clean
  Removes a virtual IP address from the interface where the virtual IP address is configured.
- Open
  Initializes the setup that the agent uses to start in a clean state.
- Close
  Cleans up the setup that the agent uses.

State definitions

- ONLINE
  Indicates that the specified IP address is assigned to the device.
- OFFLINE
  Indicates that the specified IP address is not assigned to the device.
- UNKNOWN
  Indicates that the agent can not determine the state of the resource. This may be due to an incorrect configuration.

Attributes

Table 3-5 Required attributes

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

Table 3-6 Optional attributes

Optional attribute	Description
IfconfigTwice	Causes an IP address to be configured twice using an `ifconfig up-down-up` sequence. Increases the probability of gratuitous ARP requests (generated by `ifconfig up`) to reach clients. Type and dimension: integer-scalar
NetMask	The netmask associated with the IP address of the resource. Specify the value of the netmask in decimal (base 10) or hexadecimal (base 16). **Note:** Symantec recommends that you specify a netmask for each virtual interface. Type and dimension: string-scalar
Options	Options for the `ifconfig` command. Type and dimension: string-scalar Example: "broadcast 192.203.15.255"

Resource type definition

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

Sample configuration: IPMultiNIC and MultiNICA

See "Sample configurations" on page 59.

MultiNICA agent

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

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

Agent function

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

State definitions

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

Attributes

Table 3-7 Required attributes

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

Table 3-8 Optional attributes

Optional attribute	Description
ArpDelay	Number of seconds to sleep between configuring an interface and sending out a broadcast to inform routers about the base IP address. Type and dimension: integer-scalar Default: 1
HandshakeInterval	Computes the maximum number of attempts the agent makes either to ping a host (listed in the NetworkHosts attribute) when it fails over to a new NIC, or to ping the default broadcast address (depending on the attribute configured) when it fails over to a new NIC. To prevent spurious failovers, the agent must try to contact a host on the network several times before marking a NIC as FAULTED. Increased values result in longer failover times, whether between the NICs or from system to system in the case of FAULTED NICs. Type and dimension: integer-scalar Default: 20 This is the equivalent to two attempts.

Table 3-8 Optional attributes

Optional attribute	Description
IfconfigTwice	Causes an IP address to be configured twice, using an `ifconfig up-down-up` sequence. Increases the probability of gratuitous ARP requests (caused by `ifconfig up`) to reach clients. Type and dimension: integer-scalar
NetworkHosts	List of hosts on the network that are pinged to determine if the network connection is alive. Enter the IP address of the host, instead of the host name, to prevent the monitor from timing out. DNS can cause the ping to hang. If more than one network host is listed, the monitor returns online if at least one of the hosts is alive. Type and dimension: string-vector Example: "166.93.2.1", "166.97.1.2"
NetMask	Netmask for the base IP address. You can specify the value of NetMask in decimal (base 10) or hexadecimal (base 16). **Note:** Symantec recommends that you specify a netmask for each virtual interface. Type and dimension: string-scalar
Options	The `ifconfig` options for the base IP address. Type and dimension: string-scalar Example: "broadcast 192.203.15.255"
PingOptimize	Number of monitor cycles to detect if the configured interface is inactive. A value of 1 optimizes broadcast pings and requires two monitor cycles. A value of 0 performs a broadcast ping each monitor cycle and detects the inactive interface within the cycle. Type and dimension: integer-scalar Default: 1

Table 3-8 Optional attributes

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

Resource type definition

```
type MultiNICA (
    static str ArgList[] = { Device, NetMask, ArpDelay,
    RetestInterval, Options, RouteOptions, PingOptimize,
    MonitorOnly, IfconfigTwice, HandshakeInterval, NetworkHosts }
    static int MonitorTimeout = 300
    static int OfflineMonitorInterval = 60
    static str Operations = None
    str Device{}
    str NetMask
    int ArpDelay = 1
    int RetestInterval = 5
    str Options
    str RouteOptions
    int PingOptimize = 1
    int IfconfigTwice
    int HandshakeInterval = 20
    str NetworkHosts[]
)
```

MultiNICA notes

- 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.

  On HP-UX for example, you have one active NIC, lan0(10.128.2.5), and you configure a second NIC, lan1, as the backup NIC to lan0. The agent does not fail over from lan0 to lan1 because all ping tests are redirected through lan0 on the same subnet, making the MultiNICA monitor return an online status. Note that using ping -i does not enable the use of multiple active NICs.
- Before you start VCS, configure the primary NIC with the correct broadcast address and netmask.

Using RouteOptions

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

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

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

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

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

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

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

Sample configurations

MultiNICA and IPMultiNIC

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

The MultiNICA resource fails over only the physical IP address to the backup NIC during a failure. The logical IP addresses are configured by the IPMultiNIC agent. The resources ip1 and ip2, shown in the following example, have the Address attribute which contains the logical IP address. If a NIC fails on sysa, the physical IP address and the two logical IP addresses fails over from `lan0` to `lan3`. If `lan3` fails, the address fails back to `lan0` if `lan0` is reconnected.

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

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

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

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

ip1 requires mnic

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

```
        )

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

    ip2 requires proxy
```

About the IPMultiNICB and MultiNICB agents

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

Checklist to ensure the proper operation of MultiNICB

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

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

IPMultiNICB agent

Works with the MultiNICB agent, Configures and manages virtual IP addresses (IP aliases) on an active network device specified by the MultiNICB resource. When the MultiNICB agent reports a particular interface as failed, the IPMultiNICB agent moves the IP address to the next active interface.

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

Dependencies

IPMultiNICB resources depend on MultiNICB resources.

Requirements for IPMultiNICB

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

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

Agent functions

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

State definitions

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

Attributes

Table 3-9 Required attributes

Required attribute	Description
Address	The logical IP address that the IPMultiNICB resource must handle. Type and dimension: string-scalar Example: "192.205.10.15"
BaseResName	Name of MultiNICB resource from which the IPMultiNICB resource gets a list of working interfaces. The logical IP address is placed on the physical interfaces according to the device number information. Type and dimension: string-scalar Example: "gnic_n"

Table 3-10 Optional attributes

Optional attribute	Description
DeviceChoice	Indicates the preferred NIC where you want to bring the logical IP address online. Specify the device name or NIC alias as determined in the Device attribute of the MultiNICB resource. Type and dimension: string-scalar Default: 0 Examples: DeviceChoice = "lan0" DeviceChoice = "1"
NetMask	Netmask for the base IP address. You can specify the value of NetMask in decimal (base 10) or hexadecimal (base 16). **Note:** Symantec recommends that you specify a netmask for each virtual interface. Type and dimension: string-scalar

Resource type definition

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

Manually migrating a logical IP address

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

In the following form, the command shows the status of the interfaces for the specified MultiNICB resource.

```
# haipswitch -s MultiNICB_resname
```

In the following form, the command checks that both *from* and *to* interfaces are associated with the specified MultiNICB resource and the *to* interface is working. If not, the command aborts the operation. It then removes the IP address on the *from* logical interface and configures the IP address on the *to* logical interface. Finally it erases previous failover information created by MultiNICB for this logical IP address.

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

Sample configurations

Other sample configurations for IPMultiNICB and MultiNICB

See “IPMultiNICB and MultiNICB configuration” on page 71.

MultiNICB agent

Works with the IPMultiNICB agent. Allows IP addresses to fail over to multiple NICs on the same system before VCS attempts to fail over to another system.

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

The agent monitors the interfaces it controls by sending packets to other hosts on the network and checking the link status of the interfaces.

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

Each NIC must have its own unique and exclusive base IP address, which the agent uses as the test IP address.

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

MultiNICB uses the following criteria to determine if an interface is working:

- Interface status: The interface status as reported by driver of the interface (assuming the driver supports this feature). This test is skipped if the attribute IgnoreLinkStatus = 1.
- ICMP echo: ICMP echo request packets are sent to one of the network hosts (if specified). Otherwise, the agent uses ICMP broadcast and caches the sender of the first reply as a network host. While sending and receiving ICMP packets, the IP layer is completely bypassed.

The MultiNICB agent writes the status of each interface to an export information file, which other agents (like IPMultiNICB) or commands (like `haipswitch`) can read.

Failover and failback

During an interface failure, the MultiNICB agent fails over all logical IP addresses to a working interface under the same resource. The agent remembers the first physical interface from which an IP address was failed over. This physical interface becomes the "original" interface for the particular logical IP address. When the original interface is repaired, the logical IP address fails back to it.

Agent functions

- Open

  Allocates an internal structure to store information about the resource.

- Close

  Frees the internal structure used to store information about the resource.

- Monitor

  Checks the status of each physical interface. Writes the status information to the export information file for IPMultiNICB resources to read it.

  Performs failover. Performs failback if the value of the Failback attribute is 1.

State definitions

- ONLINE

  Indicates that one or more of the network interfaces listed in the Device attribute of the resource is in working condition.

- UNKNOWN

  Indicates that the MultiNICB resource is not configured correctly.

- FAULTED

  Indicates that all of the network interfaces listed in the Device attribute failed.

Attributes

Table 3-11 Required attribute

Required attribute	Description
Device	List of NICs that you want under MultiNICB control, and the aliases of those NICs. The IPMultiNICB agent uses the NIC aliases to configure IP addresses. The IPMultiNICB agent uses these interface aliases to determine the order of the interface on which to bring the IP addresses online. Type and dimension: string-association Example: Device = { "lan0", "lan4" } Example: Device = { "lan0" = 0, "lan1" = 2, "lan2" = 3 } In this example, the MultiNICB agent uses interfaces lan0, lan1, and lan2. The MultiNICB agent passes on the associated interface aliases 0, 2, and 3 to the IPMultiNICB agent.

Table 3-12 Optional attributes

Optional attribute	Description
DefaultRouter	This is the IP address of the default router on the subnet. If specified, the agent removes the default route when the resource goes offline. The agent adds the route back when the group returns online. You must specify this attribute if multiple IP subnets exist on one host; otherwise, the packets cannot be routed properly when the subnet corresponding to the first default route goes down. Type and dimension: string-scalar Default: 0.0.0.0 Example: "192.1.0.1"
Failback	If set to 1, the virtual IP addresses are failed back to the original physical interface whenever possible. A value of 0 disables this behavior. Type and dimension: integer-scalar Default: 0

Table 3-12 Optional attributes

Optional attribute	Description
IgnoreLinkStatus	If set to 1, the agent ignores the driver-reported interface status while testing the interfaces. If set to 0, the agent reports the interface status as down if the driver-reported interface status indicates the down state. Using interface status for link testing may considerably speed up failovers. Type and dimension: integer-scalar Default: 1
LinkTestRatio	This is the ratio of total monitor cycles to monitor cycles in which the agent tests the interfaces by sending packets. At all other times, the agent tests the link by checking the "link-status" as reported by the device driver. Checking the "link-status" is a faster way to check the interfaces, but only detects cable disconnection failures. If set to 1, packets are sent during every monitor cycle. If set to 0, packets are never sent during a monitor cycle. Type and dimension: integer-scalar Default: 1 Example: 3 In this example, if the monitor agent function invokes in a numbered pattern such as 1, 2, 3, 4, 5, 6, ..., the actual packet send test is done at 3, 6, ... monitor agent functions. For LinkTestRatio=4, the packet send test is done at 4, 8, ... monitor agent functions.
NetworkHosts	List of host IP addresses on the IP subnet that are pinged to determine if the interfaces are working. NetworkHosts only accepts IP addresses to avoid DNS lookup delays. The IP addresses must be directly present on the IP subnet of interfaces (the hosts must respond to ARP requests). If IP addresses are not provided, the hosts are automatically determined by sending a broadcast ping (unless the NoBroadcast attribute is set to 1). The first host to reply serves as the ping destination. Type and dimension: string-vector Example: "192.1.0.1"

Table 3-12 Optional attributes

Optional attribute	Description
NetworkTimeout	Timeout for ARP and ICMP packets in milliseconds. MultiNICB waits for response to ICMP and ARP packets only during this time period. Assign NetworkTimeout a value in the order of tens of milliseconds (given the ICMP and ARP destinations are required to be on the local network). Increasing this value increases the time for failover. Type and dimension: integer-scalar Default: 100
NoBroadcast	If set to 1, NoBroadcast prevents MultiNICB from sending broadcast ICMP packets. (Note: MultiNICB can still send ARP requests.) If NetworkHosts are not specified and NoBroadcast is set to 1, the MultiNICB agent cannot function properly. **Note:** Symantec does not recommend setting the value of NoBroadcast to 1. Type and dimension: integer-scalar Default: 0
OfflineTestRepeat Count	Number of times the test is repeated if the interface status changes from UP to DOWN. For every repetition of the test, the next NetworkHost is selected in round-robin manner. At the end of this process, broadcast is performed if NoBroadcast is set to 0. A greater value prevents spurious changes, but also increases the response time. Type and dimension: integer-scalar Default: 3
OnlineTestRepeatC ount	Number of times the test is repeated if the interface status changes from DOWN to UP. This helps to avoid oscillations in the status of the interface. Type and dimension: integer-scalar Default: 3

Resource type definition

```
type MultiNICB (
    static int MonitorInterval = 10
    static int OfflineMonitorInterval = 60
    static int MonitorTimeout = 60
    static int Operations = None
    static str ArgList[] = { Device, NetworkHosts, LinkTestRatio,
    IgnoreLinkStatus, NetworkTimeout, OnlineTestRepeatCount,
    OfflineTestRepeatCount, NoBroadcast, DefaultRouter, Failback }
    str Device{}
    str NetworkHosts[]
    int LinkTestRatio = 1
    int IgnoreLinkStatus = 1
    int NetworkTimeout = 100
    int OnlineTestRepeatCount = 3
    int OfflineTestRepeatCount = 3
    int NoBroadcast = 0
    str DefaultRouter = "0.0.0.0"
    int Failback = 0
)
```

Trigger script

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

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

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

The sample script does the following:

- If interface changes status, it prints a message to the console, for example: MultiNICB: Device lan0 status changed from down to up.
- The script saves last IP address-to-interface name association. If any of the IP addresses has been moved, added, or removed, it prints out a message to the console, for example: MultiNICB: IP address 192.4.3.3 moved from interface lan1:1 to interface lan0:1

IPMultiNICB and MultiNICB configuration

The following is an example VCS configuration.

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

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

system north

system south

group g11 (
    SystemList = { north = 0, south = 1 }
    AutoStartList = { north, south }
    )

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

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

    ipmnicb requires mnicb
```

DNS agent

The DNS agent updates and monitors the 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 3-13 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” 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 3-14 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 3-15 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:

- HP-UX

  ```
  www.example.com.   86400  IN  CNAME  virtual1.example.com
  ```

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

Secure DNS update

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

Setting up secure updates using TSIG keys

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

To use secure updates using TSIG keys

1 Run the `dnskeygen` command with the HMAC-MD5 (`-H`) option to generate a pair of files that contain the TSIG key:

```
# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST veritas.com.
    Kveritas.com.+157+00000.key
    Kveritas.com.+157+00000.private
```

2 Open either file. The contents of the file should look similar to:

```
veritas.com. IN KEY 513 3 157 +Cdjlkef9ZTSeixERZ433Q==
```

3 Copy the shared secret (the TSIG key), which should look similar to:

```
+Cdjlkef9ZTSeixERZ433Q==
```

4 Configure the DNS server to only allow TSIG updates using the generated key.

Open the named.conf file and add these lines.

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

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

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

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

6. Save and restart the named process.

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

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

   ```
   DNS www (
   Domain = "veritas.com"
   Alias = www
   Hostname = north
   TSIGKeyFile = "/var/tsig/Kveritas.com.+157+00000.private"
   )
   ```

File share agents

This chapter contains the following:

- "About the file service agents" on page 79
- "NFS agent" on page 80
- "NFSRestart agent" on page 82
- "Share agent" on page 87

About the file service agents

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

NFS agent

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

Agent functions

- Online
  Checks if nfsd and rpc.mountd daemons are running. If they are not running, the agent starts the daemons.
- Monitor
  Monitors versions 2 and 3 of the nfsd daemons, and versions 1 and 3 of the rpc.mountd daemon. Monitors TCP and UDP versions of the daemons by sending RPC (Remote Procedure Call) calls `clnt_create` and `clnt_call` to the RPC server. If the calls succeed, the resource is reported ONLINE.
- Clean
  Terminates and restarts the nfsd and rpc.mountd daemons.

State definitions

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

Attributes

Table 4-1 Optional attributes

Optional attribute	Description
LockFileTimeout	If the group goes offline, the agents waits for the specified time before restarting nfsd and rpc.mountd daemons. Type and dimension: integer-scalar Default: 180 (lower limit: 90 seconds, upper limit: 300 seconds)

Table 4-1 Optional attributes

Optional attribute	Description
Nservers	Specifies the number of concurrent NFS requests the server can handle. Type and dimension: integer-scalar Default: 4 Example: 24
Protocol	Selects the transport protocol that the NFS server supports. Allowed values are: tcp, udp or all. If you define Protocol to equal all, the NFS server supports both protocols. Type and dimension: string-scalar Default: all Example: "tcp"

Resource type definition

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

Sample configurations

Configuration

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

NFSRestart agent

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

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

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.
  - For all NFSLock resources, copies the locks from the shared storage to the /var/statmon/sm directory if NFSLockFailover is set to 1.
  - Starts the statd and lockd daemons.
  - Starts the smsyncd daemon to copy the contents of /var/statmon/sm directory to the shared storage (LocksPathName) at regular, two second intervals.
- Monitor
  - Monitors smsyncd if LockFailOver is set to 1. Otherwise it monitors rpc.statd and rpc.lockd and restarts them if they are not running.
- Offline
  - Terminates the statd and lockd daemons to clear the lock state.
  - Terminates the nfsd and mountd daemons to close the TCP/IP connections.
  - Terminates the smsyncd daemon.
- Clean
  - Terminates the statd and lockd daemons to clear the lock state.
  - Terminates the nfsd and mountd daemons to close TCP/IP connections.
  - Terminates the smsyncd daemon.

State definitions

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

Attributes

Table 4-2 Required attributes

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

NFSRestart notes

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

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

```
    /etc/hosts
```

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

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

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

For example:

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

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

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

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

Resource type definition

```
type NFSRestart (
    static str ArgList[] = { "NFSRes:LockFileTimeout",
    LocksPathName, NFSLockFailOver  }
    str NFSRes
    str LocksPathName
    boolean NFSLockFailOver = 0
)
```

Sample configurations

```
include "types.cf"

cluster vcs_test (
        )

system sysA (
        )
```

```
system sysB (
        )

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

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

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

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

        NFS nfs1 (
                Nservers = 8
                LockFileTimeout= 360
                )

        NFSRestart nfsres1 (
                LocksPathName = "/dir2"
                NFSLockFailOver = 1
                NFSRes = nfs1
                )

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

        Share Share_dir1 (
                PathName = "/dir1"
                )

        Volume dg01_vol01 (
                Volume = vol01
                DiskGroup = dg01
                )

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

Share agent

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

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

Dependencies

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

Agent functions

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

State definitions

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

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

Resource type definition

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

Sample configurations

Configuration

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

Chapter 5

Service and application agents

This chapter contains the following agents:

- "Apache Web server agent" on page 90
- "Application agent" on page 98
- "Process agent" on page 104
- "ProcessOnOnly agent" on page 108

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

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

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

        DiskGroup Apache_dg (
                Critical = 0
                DiskGroup = apc1
                )

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

        Mount Apache_mnt (
                Critical = 0
        MountPoint = "/apache"
        BlockDevice = "/dev/vx/dsk/apc1/apcvol1"
        FSType = vxfs
        FsckOpt = "-y"
        )

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

Application agent

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/sample_app start"
StopProgram	The executable, created locally on each node, that stops the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and are separated by spaces. Type and dimension: string-scalar Example: "/usr/sbin/sample_app stop"
At least one of the following attributes: ■ MonitorProcesses ■ MonitorProgram ■ PidFiles	See “Optional attributes” on page 100.

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 Example: "/usr/sbin/sample_app force stop"

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 full command line argument displayed by the `ps -u` *`user`* `-o args | more` command for the process. Type and dimension: string-vector Example: "sample_app_process"
MonitorProgram	The executable, created locally on each node, which monitors the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and are separated by spaces. MonitorProgram can return the following VCSAgResState values: OFFLINE value is 100; online values range from 101 to 110 (depending on the confidence level); 110 equals confidence level of 100%. Any other value = UNKNOWN. Type and dimension: string-scalar Example: "/usr/sbin/sample_app_monitor all"
PidFiles	A list of PID files that contain the process ID (PID) of the processes that you want monitored and cleaned. These are application generated files. Each PID file contains one monitored PID. Specify the complete path of each PID file in the list. The process ID can change when the process restarts. If the application takes time to update the PID file, the agent's monitor script may return an incorrect result. If this occurs, increase the ToleranceLimit in the resource definition. Type and dimension: string-vector Example: "/etc/sample/sample_app.pid"

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

Sample configurations

Sample Configuration 1

In this example, configure the executable samba 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 sendmail (
    User = root
    StartProgram = "/sbin/init.d/sendmail start"
    StopPragram = "/sbin/init.d/sendmail stop"
    PidFiles = {"/etc/mail/sendmail.pid"}
)
```

Sample Configuration 2

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

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

Process agent

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 80 characters. Type and dimension: string-scalar Example: "/usr/sbin/sendmail"

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. This attribute must not exceed 80 characters. Type and dimension: string-scalar Example: "bd -q30m"

Table 5-6 Optional attributes

Optional attribute	Description
PidFile	File containing the process ID. Type and dimension: string-scalar Example: "/etc/mail/sendmail.pid"
Priority	Priority with which the process runs. Effective only when the user is root. Range is 0 to 39 where a process with a priority 0 is the highest. Type and dimension: string-scalar Default: 20 Example: "35"
UserName	The user whose ID is used to run the process. The process along with the arguments must run the context of the specified user. Type and dimension: string-scalar Default: root Example: "user1"

Resource type definition

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

Sample configurations

Configuration 1

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

Sample configuration 2

```
include "types.cf"

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

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

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

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

ProcessOnOnly agent

Starts and monitors a 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
IgnoreArgs	A flag that indicates whether monitor ignores the argument list. ■ If the value is 0, it checks the process pathname and argument list. ■ If the value is 1, it only checks for the executable pathname and ignores the rest of the argument list. Type and dimension: boolean-scalar Default: 0
PathName	Defines complete pathname to access an executable program. This path includes the program name. If a process is controlled by a script, the PathName defines the complete path to the shell. Pathname must not exceed 80 characters. Type and dimension: string-scalar Example: "/usr/sbin/sendmail"

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. Arguments must not exceed a total of 80 characters. Type and dimension: string-scalar Example: "-bd -q30m"

Resource type definition

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

Sample configurations

Configuration 1

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

Chapter 6

Infrastructure and support agents

This chapter contains the following agents:

- "NotifierMngr agent" on page 112
- "VRTSWebApp agent" on page 119
- "Proxy agent" on page 122
- "Phantom agent" on page 125
- "RemoteGroup agent" on page 127

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. **Note:** SnmpConsoles is a required attribute if SmtpServer is not specified; otherwise, SnmpConsoles is an optional attribute. Specify both SnmpConsoles and SmtpServer if desired. Type and dimension: string-association Example: "172.29.10.89" = Error, "172.29.10.56" = Information
SmtpServer	Specifies the machine name of the SMTP server. **Note:** SmtpServer is a required attribute if SnmpConsoles is not specified; otherwise, SmtpServer is an optional attribute. You can specify both SmtpServer and SnmpConsoles. Type and dimension: string-scalar Example: "smtp.your_company.com"

Table 6-2 Optional attributes

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

Table 6-2 Optional attributes

Optional attribute	Description
MessagesQueue	Size of the VCS engine's message queue. Minimum value is 30. Type and dimension: integer-scalar Default: 30
NotifierListeningPort	Any valid, unused TCP/IP port number. Type and dimension: integer-scalar Default: 14144
SmtpFromPath	Set to a valid email address, if you want the notifier to use a custom email address in the FROM: field. Type and dimension: string-scalar Example: "usera@example.com"
SmtpRecipients	Specifies the email address where SMTP sends information and the severity level of the messages. The severity levels of messages are: Information, Warning, Error, and SevereError. Specifying a given severity level for messages indicates that all messages of equal or higher severity are received. **Note:** SmtpRecipients is a required attribute if you specify SmtpServer. Type and dimension: string-association Example: "james@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 set the SmtpVrfyOff to `1` in order for the SmtpReturnPath value to take effect. Type and dimension: string-scalar Example: "usera@example.com"
SmtpServerTimeout	This attribute represents the time in seconds notifier waits for a response from the mail server for the SMTP commands it has sent to the mail server. If you notice that the mail server is taking a longer duration to reply back to the SMTP commands sent by notifier, you can increase this value. Type and dimension: integer-scalar Default: 10
SmtpServerVrfyOff	Set this value to `1` if your mail server does not support SMTP VRFY command. If you set this value to `1,` the notifier does not send a SMTP VRFY request to the mail server specified in the SmtpServer attribute while sending emails. Type and dimension: boolean-scalar Default: 0
SnmpCommunity	Specifies the community ID for the SNMP manager. Type and dimension: string-scalar Default: public

Table 6-2 Optional attributes

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

Resource type definition

```
type NotifierMngr (
    static int RestartLimit = 3
    static str ArgList[] = { EngineListeningPort, MessagesQueue,
    NotifierListeningPort, SnmpdTrapPort, SnmpCommunity,
    SnmpConsoles, SmtpServer, SmtpServerVrfyOff, SmtpServerTimeout,
    SmtpReturnPath, SmtpFromPath, SmtpRecipients }
    int EngineListeningPort = 14141
    int MessagesQueue = 30
    int NotifierListeningPort = 14144
    int SnmpdTrapPort = 162
    str SnmpCommunity = "public"
    str SnmpConsoles{}
    str SmtpServer
    boolean SmtpServerVrfyOff = 0
    int SmtpServerTimeout = 10
    str SmtpReturnPath
    str SmtpFromPath
    str SmtpRecipients{}
)
```

Sample configuration

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

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

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

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

The NotifierMngr resource sets up notification for all events to the `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_en0 (
        Device  = lan0
        NetworkHosts = { "166.93.2.1", "166.97.1.2" }
        )

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

    Proxy nicproxy(
    TargetResName = "NicGrp_en0"
)

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

ntfr requires nicproxy


// resource dependency tree
//
//      group Grp1
//      {
//      NotifierMngr ntfr
//              {
//              Proxy nicproxy
//              }
//      }
```

VRTSWebApp agent

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

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

Agent functions

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

State definitions

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

Attributes

Table 6-3 Required attributes

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

Resource type definition

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

Sample configuration

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

Proxy agent

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

Sample configurations

Configuration 1

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

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

Configuration 2

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

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

Configuration

The proxy resource mirrors the state of the resource mnic on the local system; note that target resource is in grp1, and the proxy is in grp2; a target resource and its proxy cannot be in the same group.

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

        MultiNICA mnic (
            Device@sysa = { lan0 = "192.98.16.103",lan3 =
"192.98.16.103" }
            Device@sysb = { lan0 = "192.98.16.104",lan3 =
"192.98.16.104" }
            NetMask = "255.255.255.0"
            ArpDelay = 5
            Options = "broadcast 192.203.15.255"
            RouteOptions@sysa = "default 192.98.16.103 0"
```

```
                RouteOptions@sysb = "default 192.98.16.104 0"
                )
            IPMultiNIC ip1 (
                Address = "192.98.14.78"
                NetMask = "255.255.255.0"
                MultiNICResName = mnic
                Options = "broadcast 192.203.15.255"
                )

        ip1 requires mnic


    group grp2 (
            SystemList = { sysa, sysb }
            AutoStartList = { sysa }
            )
            IPMultiNIC ip2 (
                Address = "192.98.14.79"
                NetMask = "255.255.255.0"
        MultiNICResName = mnic
                Options = "mtu m"
                )
        Proxy proxy (
                TargetResName = mnic
                )
        ip2 requires proxy
```

Phantom agent

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.

Attribute

Table 6-6 Attribute for

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

Resource type definition

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

Sample configurations

Configuration 1

```
Phantom (
)
```

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 130 for more information.
- Offline
  Takes the remote service group offline.
  See the “ControlMode” on page 130 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 129.
- Clean
  If the RemoteGroup resource faults, the Clean function takes the remote service group offline.
  See the “ControlMode” on page 130 for more information.

State definitions

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

Attributes

Table 6-7 Required attributes

Required attribute	Description
IpAddress	The IP address or DNS name of a node in the remote cluster. The IP address can be either physical or virtual. When configuring a virtual IP address of a remote cluster, do not configure the IP resource as a part of the remote service group. Type and dimension: string-scalar 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 6-7 Required attributes

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

Table 6-7 Required attributes

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

Table 6-8 Optional attributes

Optional attribute	Description
DomainType	For a secure remote cluster only, enter the domain type information for the specified user. For users who have the domain type unixpwd, you do not have to set this attribute. Type: string-scalar Example: "nis", "nisplus"
BrokerIp	For a secure remote cluster only, if 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 6-9 Type-level attributes

Type level attributes	Description
OnlineRetryLimit OnlineWaitLimit ToleranceLimit MonitorInterval AutoFailover	In case of remote service groups that take a longer time to Online, Symantec recommends that you modify the default OnlineWaitLimit and OnlineRetryLimit attributes. If you expect the RemoteGroup agent to tolerate sudden offlines of the remote service group, then modify the ToleranceLimit attribute. See the *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:

- "ElifNone agent" on page 136
- "FileNone agent" on page 137
- "FileOnOff agent" on page 138
- "FileOnOnly agent" on page 140

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.

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

Numerics

802.1Q trunking 43

A

about
  Network agents 41
ACC library 96
agent
  modifying 14
agent functions 87
  Apache Web server agent 91
  Application agent 98
  DiskGroup agent 18
  DNS agent 73
  ElifNone agent 136
  FileNone agent 137
  FileOnOff agent 138
  FileOnOnly agent 140
  IP agent 44
  IPMultiNIC agent 51
  IPMultiNICB agent 62
  LVMCombo agent 31
  LVMLogicalVolume agent 25
  LVMVolumeGroup agent 28
  Mount agent 35
  MultiNICA agent 54
  MultiNICB agent 67
  NFS agent 80
  NFSRestart agent 82
  NIC agent 48
  NotifierMngr agent 112
  Phantom agent 125
  Process agent 104
  ProcessOnOnly agent 108
  Proxy agent 122
  RemoteGroup agent 128
  Volume agent 23
  VRTSWebApp agent 119
agents
  Apache Web server 90
  Application 98
  DiskGroup 18
  DNS 73
  ElifNone 136
  FileNone 137
  FileOnOff 138
  FileOnOnly 140
  IP 44
  IPMultiNIC agent 51
  IPMultiNICB agent 62
  LVMCombo agent 31
  LVMLogicalVolume 25
  LVMVolumeGroup agent 28
  Mount 35
  MultiNICA 54
  MultiNICB agent 66
  NFS 80
  NFSRestart agent 82
  NIC 48
  NotifierMngr agent 112
  Phantom 125
  Process 104
  ProcessOnOnly 108
  Proxy 122
  RemoteGroup 127
  Share 87
  Volume 23
  VRTSWebApp 119
agents, typical functions 13
Apache Web server agent
  ACC library 96
  agent functions 91
  attributes 92
  description 90
  detecting application failure 95
  sample configuration 96
  state definitions 91

Application agent
  agent functions 98
  description 98
  resource type definition 102
  sample configurations 102
  state definitions 99
  virtual fire drill 98
association dimension 15
attribute data types 14
attributes
  DiskGroup agent 20
  DNS agent 74
  ElifNone agent 136
  FileNone agent 137
  FileOnOff agent 138
  FileOnOnly agent 140
  IP agent 45
  IPMultiNIC agent 52
  IPMultiNICB agent 64
  LVMCombo agent 32
  LVMLogicalVolume agent 26
  LVMVolumeGroup agent 29
  Mount agent 37
  MultiNICA agent 55
  MultiNICB agent 68
  NFS agent 80
  NFSRestart agent 83
  NIC agent 49
  NotifierMngr agent 113
  Phantom agent 125
  Process agent 105
  Proxy agent 122
  RemoteGroup agent 129
  Share agent 88
  Volume agent 24
  VRTSWebApp agent 120
attributes, modifying 13, 14

B

boolean data types 14
bundled agents 13

C

Checklist to ensure the proper operation of MultiNICB 61
Cluster Manager (Java Console), modifying attributes 14
Cluster Manager (Web Console)
  modifying attributes 14
CNAME record 75
configuration files
  main.cf 125
  modifying 14
  types.cf 13

D

data type
  boolean 14
  string 14
data types
  integer 14
description, resources 13
dimensions
  keylist 15
  scalar 15
  vector 15
DiskGroup agent
  agent functions 18
  attributes 20
  description 18
  resource type definition 21
  sample configurations 21
  state definitions 19
  virtual fire drill 18
DNS agent 73
  agent functions 73
  attributes 74
  description 73
  resource type definition 75
  sample web server configuration 76

E

ElifNone agent
  agent functions 136
  attributes 136
  description 136
  resource type definition 136
  sample configuration 136

F

FileNone agent
  agent functions 137
  attribute 137
  description 137

resource type definition 137
sample configurations 137
FileOnOff agent
agent functions 138
attribute 138
description 138
FileOnOnly agent
agent functions 140
attribute 140
description 140
resource type definition 140
sample configuration 140

I

integer data types 14
Interface configuration 71
IP agent
agent functions 44
attributes 45
description 44
resource type definitions 47
sample configurations 47
state definitions 45
virtual fire drill 44
IPMultiNIC agent
agent functions 51
attributes 52
description 51
resource type definitions 53
sample configuration 53
state definitions 51
IPMultiNICB agent 65
agent functions 62
attributes 64
description 62
manually migrating IP address 65
requirements 62
resource type definition 65
state definitions 63

K

keylist dimension 15

L

LVMCombo agent
agent functions 31
attributes 32
description 31
resource type definition 33
sample configurations 33
state definitions 32
LVMLogicalVolume agent
agent functions 25
attributes 26
description 25
resource type definition 26
sample configurations 27
state definitions 25
LVMVolumeGroup agent
agent functions 28
attributes 29
description 28
resource type definition 29
sample configurations 29
state definitions 28

M

main.cf 13, 125
modifying
Cluster Manager (Web Console) 14
configuration files 14
modifying agents 14
monitor scenarios, DNS agent 76
Mount agent
agent functions 35, 36
attributes 37
description 35
resource type definition 39
sample configurations 39
virtual fire drill 35
MultiNICA agent
agent functions 54
attributes 55
description 54
resource type attributes 57
sample configurations 59
state definitions 54
MultiNICB agent
agent functions 67
attributes 68
description 66
resource type definition 70
state definitions 67

N

NFS agent
- agent functions 80
- attributes 80
- description 80
- resource type definition 81
- sample configurations 81
- state definitions 80

NFSRestart agent
- agent functions 82
- attributes 83
- description 82
- resource type definition 84
- sample configuration 84
- state definitions 83

NIC agent
- agent functions 48
- attributes 49
- description 48
- resource type definitions 50
- sample configurations 50
- state definitions 48
- virtual fire drill 48

NotifierMngr agent
- agent functions 112
- attributes 113
- description 112
- resource type definition 116
- sample configurations 117
- state definitions 112

O

online query 75

P

Phantom agent
- agent functions 125
- attributes 125
- description 125
- resource type definition 125
- sample configurations 125

Process agent
- agent functions 104
- attributes 105
- description 104
- resource type definition 106
- sample configurations 106
- state definitions 104
- virtual fire drill 104

ProcessOnOnly agent
- agent functions 108
- description 108
- resource type definition 110
- sample configurations 110
- state definitions 108

Proxy agent
- agent functions 122
- attributes 122
- description 122
- resource type definition 123
- sample configurations 123

R

RemoteGroup agent
- agent functions 128
- attributes 129
- description 127
- resource type definition 133
- state definitions 128

resource type definition 24
- FileNone agent 137

resource type definitions
- Application agent 102
- DiskGroup agent 21
- DNS agent 75
- ElifNone agent 136
- FileOnOnly agent 140
- IP agent 47
- IPMultiNIC agent 53
- IPMultiNICB agent 65
- LVMCombo agent 33
- LVMLogicalVolume agent 26
- LVMVolumeGroup agent 29
- Mount agent 39
- MultiNICA agent 57
- MultiNICB agent 70
- NFS agent 81
- NFSRestart agent 84
- NIC agent 50
- NotifierMngr agent 116
- Phantom agent 125
- Process agent 106
- ProcessOnOnly agent 110
- Proxy agent 123
- RemoteGroup agent 133
- Share agent 88

Volume agent 24
VRTSWebApp agent 120
resource types 13
resources
description of 13

S

sample configurations 65
Apache Web server agent 96
Application agent 102
DiskGroup agent 21
ElifNone agent 136
FileNone agent 137
FileOnOff agent 139
FileOnOnly agent 140
IP agent 47
IPMultiNIC 53
IPMultiNICB agent 65
LVMCombo agent 33
LVMLogicalVolume agent 27
LVMVolumeGroup agent 29
Mount agent 39
MultiNICA agent 59
NFS agent 81
NFSRestart agent 84
NIC agent 50
NotifierMngr agent 117
Phantom agent 125
Process agent 106
ProcessOnOnly agent 110
Proxy agent 123
Share agent 88
Volume agent 24
VRTSWebApp agent 120
sample DNS configuration 76
scalar dimension 15
secure DNS update 77
setting up secure updates using TSIG keys 77
Share agent 87
agent functions 87
attributes 88
description 87
resource type definitions 88
sample configurations 88
state definitions 87
state definitions 73
Apache Web server agent 91
Application agent 99
DiskGroup agent 19
DNS agent 73
IP agent 45
IPMultiNIC agent 51
IPMultiNICB agent 63
LVMCombo agent 32
LVMLogicalVolume agent 25
LVMVolumeGroup agent 28
Mount agent 36
MultiNICA agent 54
MultiNICB agent 67
NFS agent 80
NFSRestart agent 83
NIC agent 48
NotifierMngr agent 112
Process agent 104
ProcessOnOnly agent 108
RemoteGroup agent 128
Share agent 87
Volume agent 23
VRTSWebApp agent 119
string data type 14

T

trigger script 71
trunking 43
types.cf 13

V

VCS, resource types 13
vector dimension 15
virtual fire drill 18, 35, 44, 48, 98, 104
Volume agent
agent functions 23
attributes 24
description 23
sample configurations 24
state definitions 23
VRTSWebApp agent
agent functions 119
attributes 120
description 119
resource type definition 120
sample configuration 120
state definitions 119