Cluster Server 7.4 Bundled Agents Reference Guide - Linux

VERITAS™

Last updated: 2019-07-10

Legal Notice

Veritas Technologies LLC
500 E Middlefield Road
Mountain View, CA 94043

http://www.veritas.com

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:

https://www.veritas.com/support

You can manage your Veritas account information at the following URL:

https://my.veritas.com

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)	CustomerCare@veritas.com
Japan	CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The latest documentation is available on the Veritas website:

https://sort.veritas.com/documents

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:

infoscaledocs@veritas.com

You can also see documentation information or ask a question on the Veritas community site:

http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:

https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Chapter 1

Introducing Bundled agents

This chapter includes the following topics:

About Bundled agents

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

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

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

Agents can:

- Bring resources online.

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

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

Resources and their attributes

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

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

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

Modifying agents and their resources

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

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

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

Attributes

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

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

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

Table 1-1 Attribute data types

Data Type	Description
string	Enclose strings, which are a sequence of characters, in double quotes ("). Optionally enclose strings in quotes when they begin with a letter, and contains only letters, numbers, dashes (-), and underscores (_). A string can contain double quotes, but the quotes must be immediately preceded by a backslash. In a string, represent a backslash with two backslashes (\\).
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: 2147483647.
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{}.

IMF-aware agents

With Intelligent Monitoring Framework (IMF), VCS supports intelligent resource monitoring in addition to poll-based monitoring. IMF is an extension to the VCS agent framework.

The following are the IMF-aware agents:

- Apache HTTP server agent. See "Apache HTTP server agent" on page 202.

- Application agent. See “Application agent” on page 218.
- DiskGroup agent. See “DiskGroup agent” on page 22.
- Mount agent. See “Mount agent” on page 60.
- Process agent. See “Process agent” on page 264.

Enabling debug log messages

To help troubleshoot agent issues, you can enable debug log messages in the agent framework as well as the agents.

To enable agent framework debug log messages:

```
# hatype -modify agent_name LogDbg -add DBG_AGDEBUG DBG_AGINFO
DBG_AGTRACE
```

For example:

```
# hatype -modify Mount LogDbg -add DBG_AGDEBUG DBG_AGINFO DBG_AGTRACE
```

To enable agent-specific debug log messages:

```
# hatype -modify agent_name LogDbg -add debug_log_levels
```

For example:

```
# hatype -modify Mount LogDbg -add DBG_1 DBG_2 DBG_3 DBG_4 DBG_5 DBG_6
```

Alternatively, you can also use the following command:

```
# hatype -modify Mount LogDbg -add 1 2 3 4 5 6
```

Agent-specific debug log level information is specified in the agent’s description. For example, for information about the Mount agent, See “Debug log levels for Mount agent” on page 78.

For more information about log behavior, refer to the *VCS Administrator's Guide*.

VCS support for multi-pathing solutions

This section applies to the following agents:

- LVMLogicalVolume agent
- LVMVolumeGroup agent

VCS supports Dynamic Multi-Pathing (DMP) that is included as a part of Veritas InfoScale. Veritas does not support multi-pathing solutions that are not explicitly

listed in the hardware compatibility list (HCL). You can find the HCL on the SORT web site, under the Documentation tab. However, Veritas supports third-party solutions, which are included as a part of the operating systems.

Veritas aims to thoroughly test and support third-party and native solutions, but it is not possible to test all third-party multi-pathing applications. This is because of complex support matrix and a number of potential product combinations. Hence, Veritas does not officially support multi-pathing solutions that are not explicitly listed in the HCL. Also, advanced functionality such as I/O fencing with SCSI3-PGR is only supported with arrays and multi-pathing solutions listed in the HCL and only with Veritas InfoScale.

If you are using a third-party multi-pathing solution, Veritas understands your need of keeping data paths redundant and does not insist that you uninstall or disable the solution. Veritas does not consider third-party multi-pathing solutions as invalid and continues to troubleshoot any support issues. However, for persisting support issues related to multi-pathing solutions, you need to contact the multi-pathing vendor.

Chapter 2

Storage agents

This chapter includes the following topics:

About the storage agents

Storage agents monitor shared storage and make shared storage highly available. Storage includes shared disks, disk groups, volumes, and mounts.

DiskGroup agent

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

Note: The private disk group should not be configured in a parallel service group if the disk group is configured on the same shared disk across nodes.

For important information on this agent, See "Notes for DiskGroup agent" on page 31.

IMF awareness

The DiskGroup agent is Intelligent Monitoring Framework (IMF)-aware and uses Asynchronous Monitoring Framework (AMF) kernel driver for IMF notification. For more information about IMF and intelligent resource monitoring, refer to the *Cluster Server Administrator's Guide*.

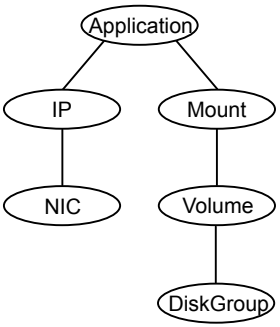
For more information about IMF-related DiskGroup agent functions, see See "Agent functions for DiskGroup agent" on page 23.

For more information about using the DiskGroup agent with IMF, see See "Using the DiskGroup agent with IMF" on page 32.

Dependencies for DiskGroup agent

The DiskGroup resource does not depend on any other resources.

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



Agent functions for DiskGroup agent

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 sets the VxVM noautoimport flag. This action allows VCS to maintain control of importing the disk group. The monitor function uses the following command to set the noautoimport flag: `# vxdg -g disk_group set autoimport=no` If IMF is enabled for the DiskGroup agent, the resource is monitored asynchronously and any change in the disk group state is immediately sent to the DiskGroup agent for appropriate action.
Clean	Terminates all ongoing resource actions and takes the resource offline—forcibly when necessary.

Info	The Info function gets information from the Volume Manager and displays the type and free size for the DiskGroup resource.
	Initiate the Info function by setting the InfoInterval timing to a value greater than 0.
	In the following example, the Info function executes every 60 seconds:
	`# haconf -makerw`
	`# hatype -modify DiskGroup InfoInterval 60`
	Enter the following command to retrieve information about the DiskType and FreeSize of the DiskGroup resource:
	`# hares -value diskgroupres ResourceInfo`
	Output includes:
	`DiskType sliced`
	`FreeSize 35354136`
	The value specified is in kilo bytes.

Action	Different actions follow: ■ license.vfd Checks for valid Veritas Volume manager license–if one is not found use the vxlicinst utility to install a valid license key. ■ disk.vfd Checks if all disks in diskgroup are visible on host—if it fails, check if the path to disks exists from the host and check if LUN masking and zoning are set properly. ■ udid.vfd Checks the UDID (unique disk identifiers) of disks on the cluster nodes—if it fails, ensure that the disks that are used for the disk group are the same on all cluster nodes. ■ verifyplex.vfd Checks if the number of plexes on each site for the Campus Cluster setup are set properly—if it fails, check that the sites, disks, and plexes are set properly for a Campus Cluster setup. ■ volinuse Checks if open volumes are in use or file systems on volumes that are mounted outside of VCS configuration. See "High availability fire drill" on page 31.
imf_init	Initializes the agent to interface with Intelligent monitoring framework (IMF). The function runs when the agent starts up.
imf_getnotification	Waits for notification about disk group state changes. The function runs after the agent initializes with IMF. The function waits for notification. Upon receiving notification, the agent takes action on the resource.
imf_register	Registers the resource entities, which the agent must monitor using IMF. The function runs for each resource after the resource goes into a steady state, either online or offline.

State definitions for DiskGroup agent

ONLINE	Indicates that the disk group is imported.
OFFLINE	Indicates that the disk group is not imported.
FAULTED	Indicates that the disk group has unexpectedly deported or become disabled.

UNKNOWN	Indicates that a problem exists either with the configuration or the ability to determine the status of the resource. One cause of this state is when I/O fencing is not configured—the cluster level attribute UseFence is not set to "SCSI3" but the Reservation attribute value is "SCSI3".

Attributes for DiskGroup agent

Table 2-1 Required attributes

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

Table 2-2 Optional attributes

Optional attributes	Description
MonitorReservation	If the value is 1 and SCSI-3 fencing is used, 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 **Note:** If the MonitorReservation attribute is set to 0, the value of the cluster-wide attribute UseFence is set to SCSI3, and the disk group is imported without SCSI reservation, then the monitor agent function takes the service group containing the disk group resource offline.

Table 2-2 Optional attributes *(continued)*

Optional attributes	Description
PanicSystemOnDGLoss	Determines whether to panic the node if the disk group becomes disabled or monitor program times out. A loss of storage connectivity can cause the disk group to become disabled. VxVM commands not responding properly can cause monitor program to timeout. **Note:** System administrators may want to set a high value for FaultOnMonitorTimeout to increase system tolerance. This attribute accepts the following values: ■ 0: Do not halt the system ■ 1: halt the system if either disk group goes into disabled state or the disk group resource faults due to monitor timeout ■ 2: halt the system if disk group goes into disabled state ■ 3: halt the system if disk group resource faults due to monitor timeout If the value of the attribute is 0 and the disk group becomes disabled, the following occurs: ■ If the cluster has I/O fencing enabled, the DiskGroup resource is marked as FAULTED. This state results in the agent attempting to take the service group offline. As part of bringing the DiskGroup resource offline, the agent attempts to deport the disabled disk group. Even if disabled disk group fails to deport, the DiskGroup resource enters the FAULTED state. This state enables the failover of the service group that contains the resource. To fail back the DiskGroup resource, manually deport the disk group after restoring storage connectivity. ■ If the cluster does not use I/O fencing, a message is logged and the resource is reported ONLINE. The resource is reported ONLINE so that it does not fail over, which ensures data integrity.

Table 2-2 Optional attributes *(continued)*

Optional attributes	Description
PanicSystemOnDGLoss (Continued)	**Note:** The PanicSystemOnDGLoss attribute does not depend on the MonitorReservation attribute. **Note:** If PanicSystemOnDGLoss is set to non-zero value, the system panic is initiated using the `poweroff -nf` command. This command halts the system. An administrator needs to bring up the system. Type and dimension: integer-scalar Default: 0
StartVolumes	If value of this attribute is 1, the DiskGroup online function starts all volumes belonging to that disk group after importing the group. **Note:** If the Veritas Volume Manager default autostartvolumes at system level is set to on, all the volumes of the disk group is started as a part of the import disk group. Type and dimension: boolean-scalar Default: 1
StopVolumes	If value is 1, the DiskGroup offline function stops all volumes belonging to that disk group before it deports the disk group. Type and dimension: boolean-scalar Default: 1

Table 2-2 Optional attributes *(continued)*

Optional attributes	Description
UmountVolumes	This attribute enables the DiskGroup resource to forcefully go offline even if open volumes are mounted outside of VCS control. When the value of this attribute is 1 and the disk group has open volumes, the following occurs: ■ The agent attempts to unmount the file systems on open volumes. If required, the agent attempts to kill all VCS managed and un-managed applications using the file systems on those open volumes. ■ The agent attempts to forcefully unmount the file systems to close the volumes. Type and dimension: integer-scalar Default: 0
Reservation	Determines if you want to enable SCSI-3 reservation. This attribute can have one of the following three values: ■ ClusterDefault—The disk group is imported with SCSI-3 reservation if the value of the cluster-level UseFence attribute is SCSI3. If the value of the cluster-level UseFence attribute is NONE, the disk group is imported without reservation. ■ SCSI3—The disk group is imported with SCSI-3 reservation if the value of the cluster-level UseFence attribute is SCSI3. ■ NONE—The disk group is imported without SCSI-3 reservation. To import a disk group with SCSI-3 reservations, ensure that the disks of the disk group are SCSI-3 persistent reservation (PR) compliant. Type and dimension: string-scalar Default: ClusterDefault Example: "SCSI3"

Table 2-2 Optional attributes *(continued)*

Optional attributes	Description
ClearClone	If the value of this attribute is 1, the disk group is imported with the '-c' option. While importing the disk group, this option clears the "clone" and "udid_mismatch" flags from the disks of the disk group and also updates the UDID, if required. For more information about the '-c' option, refer to the *VxVM manual page*. **Note:** For hardware cloning devices, do not set this attribute to 1. Type and dimension: boolean-scalar Default: 0
DGOptions	Specifies the options for the `vxdg` import command. The agent uses this attribute only while bringing a DiskGroup resource online. For more information, see the *vxdg(1m) manual page*. Type and dimension: string-scalar Example: "-o noautostart -o updateid"

Table 2-3 Internal attributes

Attribute	Description
tempUseFence	This attribute is reserved for internal use only.

Table 2-3 Internal attributes *(continued)*

Attribute	Description
NumThreads	The number of threads that are used within the agent process for managing resources. This number does not include the number of threads that are used for other internal purposes. Setting the NumThreads attribute to a higher value may decrease the time required to go online or the time required to monitor a large number of DiskGroup resources. Type and dimension: static integer-scalar Default: 1 **Note:** If there are many DiskGroup resources and if the resources are taking more time to come online, consider increasing the NumThreads attribute to a value greater than 1.

Resource type definition

The resource definition for this agent on Linux follows:

```
type DiskGroup (
static keylist SupportedActions = { "license.vfd", "disk.vfd", "udid.vfd",
"verifyplex.vfd", checkudid, numdisks, campusplex, volinuse,
joindg, splitdg, getvxvminfo }
static int OnlineRetryLimit = 1
static str ArgList[] = { DiskGroup, StartVolumes, StopVolumes, MonitorOnly,
MonitorReservation, tempUseFence, PanicSystemOnDGLoss,
UmountVolumes, Reservation, ConfidenceLevel, ClearClone }
static str IMFRegList[] = { DiskGroup, Reservation }
static int IMF{} = { Mode = 3, MonitorFreq = 5, RegisterRetryLimit = 3 }
str DiskGroup
boolean StartVolumes = 1
boolean StopVolumes = 1
static int NumThreads = 1
boolean MonitorReservation = 0
temp str tempUseFence = INVALID
int PanicSystemOnDGLoss = 0
int UmountVolumes = 0
str Reservation = ClusterDefault
```

```
boolean ClearClone = 0
)
```

Notes for DiskGroup agent

The DiskGroup agent has the following notes:

- High availability fire drill
- Using volume sets
- Setting the noautoimport flag for a disk group
- Using the DiskGroup agent with IMF

High availability fire drill

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

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

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

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

Using volume sets

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

See the Mount agent description for more information.

Setting the noautoimport flag for a disk group

VCS requires that the noautoimport flag of an imported disk group be explicitly set to true. This value enables VCS to control the importation and deportation of disk groups as needed when bringing disk groups online and taking them offline.

To check the status of the noautoimport flag for an imported disk group

- # **vxprint -l *disk_group* | grep *noautoimport***

  If the output from this command is blank, the noautoimport flag is set to false and VCS lacks the necessary control.

The Monitor function changes the value of the VxVM noautoimport flag from off to on. It changes the value instead of taking the service group offline. This action allows VCS to maintain control of importing the disk group.

The following command changes the autoimport flag to false:

```
# vxdg -g disk_group set autoimport=no
```

Configuring the Fiber Channel adapter

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

For more information, refer to the following guides:

- *Dynamic Multi-Pathing Administrator's Guide*
- *Fiber Channel Adapter's Configuration Guide*

Using the DiskGroup agent with IMF

Considerations to use the DiskGroup agent with IMF:

- You can either set the MonitorFreq to 0 or a high value. Setting the value of the MonitorFreq key to a high value ensures that the agent does not run the monitor function frequently. Setting the MonitorFreq key to 0 disables the traditional monitoring while IMF monitoring is in progress. Traditional monitoring is done after receiving the notification for a resource.

  However, if the disk group is configured with reservation and value of the MonitorReservation attribute is set to 1, then set the MonitorFreq key value to the frequency at which you want the agent to run the monitor function, to verify the reservation on the disk group.

Sample configurations for DiskGroup agent

DiskGroup resource configuration

Sample configuration of the DiskGroup resource:

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

Debug log levels for DiskGroup agent

The DiskGroup agent uses the following debug log levels:

DBG_1, DBG_3, DBG_4

DiskGroupSnap agent

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

Note: The DiskGroupSnap agent requires the Global Cluster Option (GCO) license enabled on all systems in the cluster.

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

You must define the DiskGroupSnap agent in a separate FireDrill service group which is similar to the Application service group. The FireDrill service group might contain resources similar to the Application service group, for example Mount, Application, and so on.

The FireDrill service group must also contain a resource of type DiskGroupSnap such that the Mount resource depends on the DiskGroupSnap resource. The main DiskGroup must contain multiple sites registered in it with the value of the "siteconsistent" attribute set to on.

When the DiskGroupSnap agent goes online, the agent detaches one of the sites from the main DiskGroup and imports the detached site on the fire drill host as an independent DiskGroup with a different name. The volumes on the DiskGroup are also imported and mounted with same names on the fire drill host.

The DiskGroupSnap agent provides Gold and Bronze configurations for the fire drill, which can be specified using the agent's FDType attribute. The configuration

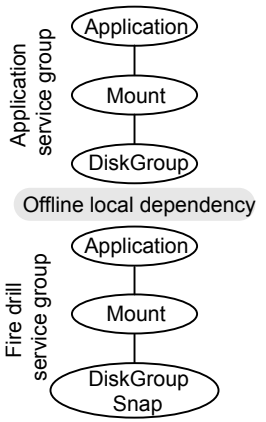
decides the site to be detached from the DiskGroup for fire drill testing. The Gold configuration is the default option in which the agent selects a site from the DiskGroup that is neither the local VxVM site nor the site on which the DiskGroup is online. With the Gold configuration, you can also specify an alternate site to detach through the agent's FDSiteName attribute. With the Bronze configuration, the agent uses the local VxVM site name as the site to detach from the DiskGroup.

For important information about this agent, See “Notes for DiskGroupSnap agent” on page 37.

Dependencies for DiskGroupSnap agent

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

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



Agent functions for DiskGroupSnap agent

Online — Verifies that the application’s disk group is in a valid campus cluster configuration. It detaches the site that the value of the FDSiteName attribute specifies. It then creates another disk group to be used for the fire drill on the detached site. After the completion of Online function, the agent creates a lock file in the lock directory (/var/VRTSvcs/lock) to indicate that the resource is online.

Offline	This re-attaches the site that the value of the FDSiteName attribute specifies back to the application's disk group. After the completion of Offline function the agent removes the lock file from the lock directory (/var/VRTSvcs/lock) to indicate that the resource is Offline.
Monitor	Monitors the DiskGroupSnap resource by checking the existence of the Lock file in /var/VRTSvcs/lock directory..
Clean	Takes the DiskGroupSnap resource offline.
Open	If the DiskGroupSnap resource has a parent resource that is not ONLINE, then it deletes the online lock file of the DiskGroupSnap resource. This marks the DiskGroupSnap resource as OFFLINE.

State definitions for DiskGroupSnap agent

ONLINE	The DiskGroupSnap resource functions normally.
OFFLINE	The DiskGroupSnap resource is not running.
UNKNOWN	A configuration error exists.
FAULTED	The DiskGroupSnap resource is taken offline unexpectedly outside of VCS control.

Attributes for DiskGroupSnap agent

Table 2-4 Required attributes

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

Table 2-4 Required attributes *(continued)*

Required attribute	Description
FDType	Specifies the configuration to be used for the fire drill. The possible values for this attribute are: ■ Bronze ■ Gold (default) The Bronze configuration uses the local host's VxVM site name as the site to be detached from the DiskGroup. This action leaves the DiskGroup vulnerable to site disaster since a copy of the production volume might not be available when the fire drill is in progress. In the Gold configuration there are at least three copies of the parent volume available on different sites, hence, even after detaching one site the volume is not vulnerable to site disaster while the fire drill is in progress.

Table 2-5 Optional attributes

Optional attribute	Description
FDSiteName	The unique VxVM site name tag for the fire drill disks. The value of this attribute is used in conjunction with the FDType attribute and it must be set to one of the sites registered in the main DiskGroup. ■ When FDType is set to the Bronze configuration, the value of FDSiteName should either be empty or the name of the local host VxVM site for the fire drill host. ■ When FDType is set to the Gold configuration, FDSiteName identifies a site in the DiskGroup to detach as a part of the fire drill. If FDSiteName is left blank, the agent will choose a site to detach based on the DiskGroup configuration. The agent chooses a site name from the DiskGroup which is neither the production server's site name nor the fire drill host's site name. Table 2-6 shows the possible values of the attributes FDType and FDSiteName and the decision taken by the agent.

Consider a configuration where the Production DiskGroup contains three sites: A, B, and C, and the Application service group is online on a node with local VxVM site ID is A. Fire drill is being done on another node Application service group is online on a node where local VxVM site ID is B.

Table 2-6 Example FDType configurations

FDType	Bronze			Gold/Empty		
FDSitename	Empty	B	C	Empty	B	C
Result	Use B as the site to detach and proceed	Detach site B from DiskGroup	Error	Check if there is another site other than A and B and select it. Else, it is an error	Error	Detach site C from the DiskGroup

Notes for DiskGroupSnap agent

The DiskGroupSnap agent has the following notes:

- See "Fire drill configuration after upgrading VCS" on page 37.
- See "Configuring the SystemZones attribute for the fire drill service group" on page 37.
- See "Configuring the FireDrill service group" on page 38.
- See "Adding the ReuseMntPt attribute to the ArgList attribute for the Mount agent type" on page 38.
- See "Configuration considerations" on page 39.
- See "Agent limitations" on page 40.

Fire drill configuration after upgrading VCS

After upgrading VCS from any earlier version to 6.0, delete all resources of type DiskGroupSnap and recreate them again using the new definitions of the attributes. Failure to perform this step might result in an unexpected behavior of the agent.

Configuring the SystemZones attribute for the fire drill service group

You must assign the local system values to the SystemZones attribute of the application's service group. You set these values so that the service group fails over in the same zone before it tries to fail over across zones.

For more information about campus cluster setup, refer to the *Cluster Server Administrator's Guide*.

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

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

Configuring the FireDrill service group

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

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

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

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

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

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

1. Make the configuration read and write.

   ```
   # haconf -makerw
   ```

2. Add the ReuseMntPt attribute to the ArgList attribute.

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

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

   ```
   # hares -modify firedrill_mount_resource_name ReuseMntPt 1
   ```

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

```
# hares -modify original_mount_resource_name ReuseMntPt 1
```

5 Make the configuration read only.

```
# haconf –dump –makero
```

Configuration considerations

Keep the following recommendations in mind:

- You must install Veritas Volume Manager 5.1 or later with the FMR license and the Site Awareness license.
- Do not bring the DiskGroupSnap resource online in the SystemZone where the application service group is online.
- Make sure that the firedrill service group and the application service group both use the same values for the SystemZones attribute.
- Do not use Volume resources in the firedrill service group. The DiskGroupSnap agent internally uses the `vxvol` command to start all the volumes in the firedrill disk group.
- In large setups, you may need to tweak the various timer values so that the timers do not time out while waiting for VxVM commands to complete. The timers you need to tweak are the OfflineTimeout for the DiskGroupSnap resource and MonitorInterval and ActionTimeout for the associated DiskGroup resource, for example:

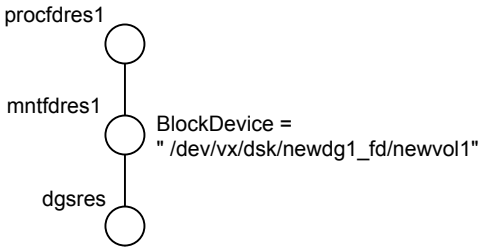
```
# haconf –makerw
# hares -override dgsres OfflineTimeout
# hares -modify dgsres OfflineTimeout 600
# hares -override dgres MonitorInterval
# hares -modify dgres MonitorInterval 1200 (this has to be twice
  the value intended for ActionTimeout below)
# hares -override dgres ActionTimeout
# hares -modify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

- When you create the firedrill service group, in general use the same attribute values that you use in the application service group.

The BlockDevice attribute of the Mount resource changes between the application service group and the firedrill service group. In the BlockDevice path, you must append an _fd to the disk group name portion, for example, /dev/vx/dsk/newdg1/newvol1 becomes /dev/vx/dsk/newdg1_fd/newvol1.
See Figure 2-3 on page 40. shows the changes to resource values for the firedrill service group; note that the Volume resource is not included.

- Before commencing the fire drill, make sure that all the sites registered in the application DiskGroup are in ACTIVE state.

Figure 2-3 Sample resource values for a DiskGroupSnap resource



Agent limitations

The following limitations apply to the DiskGroupSnap agent:

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

Resource type definition for DiskGroupSnap agent

The resource type definition for this agent follows:

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

Sample configurations for DiskGroupSnap agent

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

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

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

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

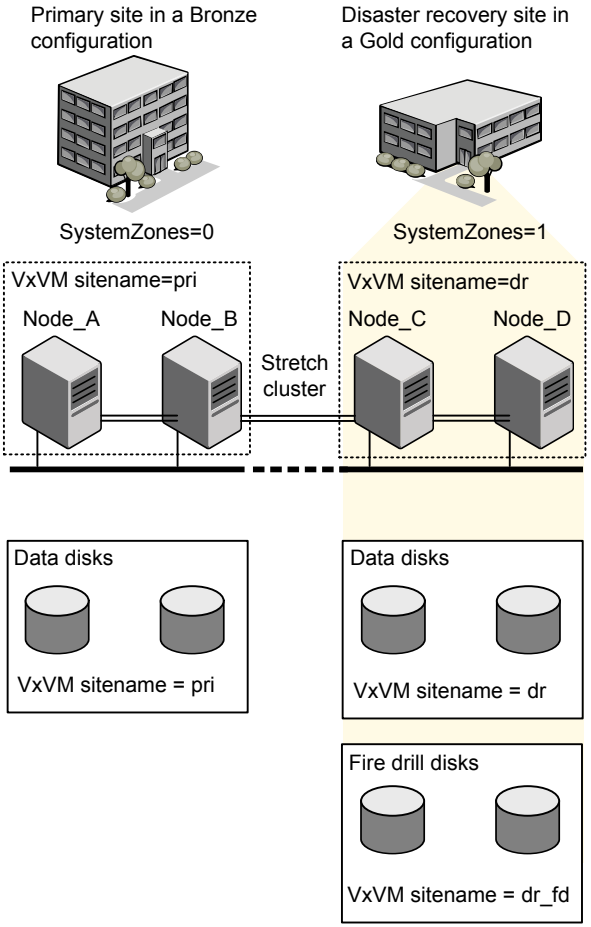
The Disaster Recovery site is in a Gold configuration as it has dedicated fire drill disks at the site. For the FDSiteName attribute, use the VxVM site tag given to the fire drill disks. For this configuration, the FDSiteName attribute values for the nodes at the Disaster recovery site follow:

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

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

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

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



Typical main.cf configuration for DiskGroupSnap agent

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

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

```
DiskGroupSnap dgsres (
TargetResName = dgres
        )

FDtype = "Gold"
        FDSiteName @Node_A = pri
        FDSiteName @Node_B = pri
        FDSiteName @Node_C = dr_fd
        FDSiteName @Node_D = dr_fd
        )

    Mount mntfdres1 (
        MountPoint = "/dgsfs1"
        BlockDevice = "/dev/vx/dsk/newdg1_fd/newvol1"
        FSType = vxfs
        FsckOpt = "-y"
        ReuseMntPt = 1
        )

    Mount mntfdres2 (
        MountPoint = "/dgsfs2"
        BlockDevice = "/dev/vx/dsk/newdg1_fd/newvol2"
        FSType = vxfs
        FsckOpt = "-y"
        ReuseMntPt = 1
        )

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

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

    requires group dgsg offline local

    mntfdres1 requires dgsres
    mntfdres2 requires dgsres
```

```
procfdres1 requires mntfdres1
procfdres2 requires mntfdres2
```

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

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

DiskGroup dgres (
     DiskGroup = newdg1
     )

Mount mntres1 (
     MountPoint = "/dgsfs1"
     BlockDevice = "/dev/vx/dsk/newdg1/newvol1"
     FSType = vxfs
     FsckOpt = "-y"
     ReuseMntPt = 1
     )

Mount mntres2 (
     MountPoint = "/dgsfs2"
     BlockDevice = "/dev/vx/dsk/newdg1/newvol2"
     FSType = vxfs
     FsckOpt = "-y"
      ReuseMntPt = 1
     )

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

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

mntres1 requires dgres
mntres2 requires dgres
```

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

Sample main.cf of DiskGroupSnap with Oracle resource

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

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

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

        IP fd_oraip (
        Device = etho
        Address = "10.198.95.191"
        NetMask = "255.255.255.0"
                    )

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

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

        NIC fd_oranic (
```

```
        Device = eth0
        NetworkHosts = { "10.198.95.1" }
        )

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

  Oracle fd_Ora_01 (
        Owner = oracle
        Home = "/opt/oracle/ora_home"
        Sid = Ora_01
        )
requires group oragrp offline local
fd_LSNR requires fd_Ora_01
fd_LSNR requires fd_oraip
fd_Ora_01 requires fd_archmnt
fd_Ora_01 requires fd_datamnt
fd_archmnt requires dgres
fd_datamnt requires dgres
fd_oraip requires fd_oranic

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

  DiskGroup oradg_res (
        DiskGroup = oradg
        )

  IP Node_A4vip (
       Device = eth0
       Address = "10.198.95.192"
       Netmask = "255.255.252.0"
        )

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

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

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

        NIC nic_Node_A4vip (
                Device = eth0
                )

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

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

        Volume arch_vol (
                Volume = archive_vol
                DiskGroup = oradg
                )

        Volume data_vol (
                Volume = data_vol
                DiskGroup = oradg
                )
LSNR requires Ora_01
LSNR requires Node_A4vip
Ora_01 requires arch_mnt
Ora_01 requires data_mnt
arch_mnt requires arch_vol
arch_vol requires oradg_res
data_mnt requires data_vol
```

```
data_vol requires oradg_res
Node_A4vip requires nic_Node_A4vip
```

Debug log levels for DiskGroupSnap agent

The DiskGroupSnap agent uses the following debug log levels:

DBG_1

Volume agent

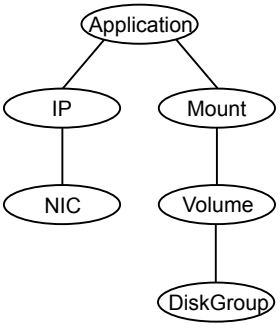
The Volume agent brings online, takes offline, and monitors a Veritas Volume Manager (VxVM) volume. Use the agent to make a volume highly available.

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

Dependencies for Volume agent

Volume resources depend on DiskGroup resources.

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



Agent functions for Volume agent

Online	Uses the `vxrecover` command to start the volume.
Offline	Uses the `vxvol` command to stop the volume.
Monitor	Attempts to read a block from the raw device interface to the volume to determine if the volume is online, offline, or unknown.

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

State definitions for Volume agent

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

Attributes for Volume agent

Table 2-7 Required attributes

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

Table 2-8 Internal attribute

Optional attribute	Description
NumThreads	Number of threads used within the agent process for managing resources. This number does not include threads used for other internal purposes. Setting the NumThreads attribute to a higher value may decrease the time required to go online or the time required to monitor a large number of DiskGroup resources. If there are many DiskGroup resources and if the resources are taking more time to come online, consider increasing the NumThreads attribute to a value greater than 1. Veritas recommends that you should not modify this attribute. Default: 1

Resource type definition for Volume agent

The resource type definition for this agent follows:

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

Sample configuration for Volume agent

The sample configuration for the Volume agent follows:

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

Debug log levels for Volume agent

The Volume agent uses the following debug log levels:

DBG_1, DBG_3, DBG_5

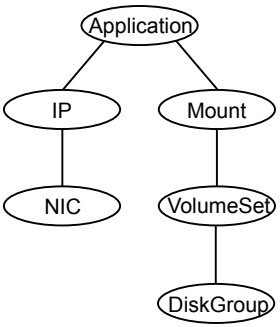
VolumeSet agent

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

Dependencies for VolumeSet agent

VolumeSet resources depend on DiskGroup resources.

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



Agent functions for VolumeSet agent

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

State definitions for VolumeSet agent

ONLINE	Indicates that all the volumes in the volume set are started and that I/O is permitted for all the volumes.
OFFLINE	Indicates that at least one of the volume is not started in the volume set and that I/O is not permitted for that volume.

FAULTED	Indicates the volumes that are inside the volume set have stopped unexpectedly and that I/O is not permitted.
UNKNOWN	Indicates that the agent could not determine the state of the resource or that the resource attributes are configured incorrectly.

Attributes for VolumeSet agent

Table 2-9 Required attributes

Required attribute	Description
DiskGroup	The name of the disk group that contains the volume set. Type and dimension: string-scalar Example: "DG1"
VolumeSet	The name of the volume set from the disk group that you specified in the DiskGroup attribute. Type and dimension: string-scalar Example: "DG1VolSet1"

Resource type definition for VolumeSet agent

```
type VolumeSet (
    static str ArgList[] = { DiskGroup, VolumeSet }
    str VolumeSet
    str DiskGroup
)
```

Sample configurations for VolumeSet agent

This sections contains sample configurations for this agent.

A configured VolumeSet that is dependent on a DiskGroup resource

The VolumeSet's shared_vset3 resource is configured and is dependent on DiskGroup resource with a shared diskgroup.

```
VolumeSet sharedg_vset3 (
VolumeSet = vset3
```

```
DiskGroup = sharedg
)
```

Agent notes for VolumeSet agent

This sections contains notes about this agent.

Inaccessible volumes prevent the VolumeSet agent from coming online

The VolumeSet agent does not come online if any volume is inaccessible in its volume set.

To remove a volume from volume set

- Enter the following commands to remove a volume from a volume set mounted on mountpoint.

```
# fsvoladm remove mountpoint volume_name
# vxvset -g diskgroup rmvol volumeset volume_name
```

Debug log levels for VolumeSet agent

The VolumeSet agent uses the following debug log levels:

DBG_1, DBG_4

LVMLogicalVolume agent

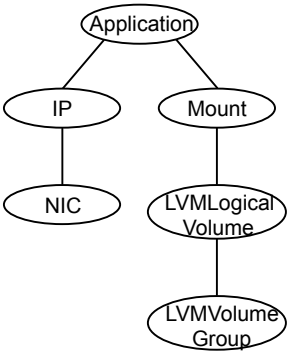
The LVMLogicalVolume agent brings online, takes offline, and monitors a Logical Volume Manager (LVM2) volume. This agent uses LVM2 commands. You can use this agent to make volume groups and logical volumes highly available and to monitor them.

See "VCS support for multi-pathing solutions" on page 19.

Dependencies

LVMLogicalVolume resources depend on LVMVolumeGroup resources.

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



Agent functions

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

State definitions

The state definitions for this agent follow:

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

Attributes

Table 2-10 Required attributes

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

Resource type definition

The resource definition for this agent on Linux follows:

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

Sample configuration

In this example, the logical volumes are mounted at a mount point.

```
Mount mnt_lvmvol01 (
     MountPoint = "/mnt/lvmvol01"
     BlockDevice = "/dev/mapper/lvmvg01-lvmvol01"
     FSType = "reiserfs"
     FsckOpt = "-y"
     )
  LVMLogicalVolume lvmvol01 (
     LogicalVolume = lvmvol01
     VolumeGroup = lvmvg01
     )
  LVMVolumeGroup lvmvg01 (
     VolumeGroup = lvmvg01
```

```
    )
mnt_lvmvol01 requires lvmvol01
lvmvol01 requires lvmvg01
```

Debug log levels

The LVMLogicalVolume agent uses the following debug log levels:

DBG_1, DBG_3, DBG_5

LVMVolumeGroup agent

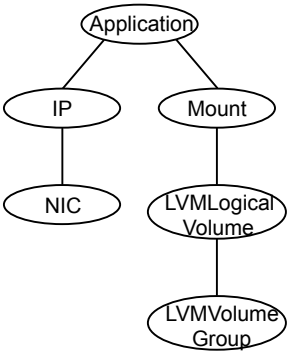
The LVMVolumeGroup agent brings online, takes offline, and monitors a Logical Volume Manager (LVM2) volume group. This agent uses LVM2 commands. You can use this agent to make volume groups and logical volumes highly available and to monitor them.

See "VCS support for multi-pathing solutions" on page 19.

Dependencies for LVMVolumeGroup agent

Veritas recommends using LVM tagging with Dynamic Multi-Pathing.

Figure 2-8 Sample service group that includes a LVMVolumeGroup resource



Agent functions for LVMVolumeGroup agent

The agent functions for this agent follow:

Online	Imports the volume group using the `vgimport` command.
Offline	Exports the volume group using the `vgexport` command.

Monitor	Determines if the volume group is online or offline using the `vgdisplay` command.
Clean	Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.
Action	Different action agent functions follow: ■ volinuse Checks if the specified volume is mounted outside of VCS control or not. In case it is mounted outside of VCS control, the action returns "FAILURE", otherwise it returns "SUCCESS".

State definitions for LVMVolumeGroup agent

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

Attributes for LVMVolumeGroup agent

Table 2-11 Required attributes

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

Table 2-12 Optional attributes

Optional attribute	Description
StartVolumes	If the value of this attribute is 1, the LVMVolumeGroup online function imports the group. It then starts all the volumes that belong to that volume group. Type and dimension: boolean-scalar Default: 0

Table 2-12 Optional attributes *(continued)*

Optional attribute	Description
EnableLVMTagging	If the value of this attribute is 1, the LVMVolumeGroup online function associates an LVM tag with the volume group and then imports the group. While taking the resource offline, it removes the associated tags. You should manually configure native LVM on all the hosts to enable LVM tagging. See "Enabling volume group activation protection for Dynamic Multi-Pathing" on page 58. Type and dimension: boolean-scalar Default: 0

Resource type definition for LVMVolumeGroup agent

The resource definition for this agent on Linux follows:

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

LVMVolumeGroup agent notes

The LVMVolumeGroup agent has the following notes:

Enabling volume group activation protection for Dynamic Multi-Pathing

Notes for volume group activation protection

Enabling volume group activation protection for Dynamic Multi-Pathing

On each node in the cluster, perform the following procedure to enable activation protection for volume groups on Red Hat and SUSE systems.

To enable volume group activation protection

1. On each node in the cluster, edit the /etc/lvm/lvm.conf file and add the following line:

   ```
   tags { hosttags = 1 }
   ```

2. In the /etc/ lvm/ directory on each node in the cluster, create the node-specific lvm.conf file in the following format:

   lvm_*host_name*.conf

   host_name is the output of the `uname -n` command.

3. Add the following line to the file you created in step 2:

   ```
   activation { volume_list=node }
   ```

   node is the output of the `uname -n` command.

Notes for volume group activation protection

Review the following notes if you plan to use volume group activation protection:

- Setting hosttag=0 in /etc/lvm/lvm.conf disables the lvm tagging for all the volume groups.
- If the system's LVM configuration (/etc/lvm/lvm.conf) is modified to support LVM tagging (hosttags = 1 in the lvm.conf file), make sure that the value of EnableLVMTagging is 1 for all the configured LVMVolumeGroup resources.
  Another way to disable the LVM tagging for a particular volume group is to add that volume group name in a volume_list activation configuration (/etc/lvm/lvm_`uname –n`.conf).
  For example, if you do not want LVM tagging to be enforced for the vg11 volume group , then add the following line in /etc/lvm/lvm_`uname – n`.conf. . activation { volume_list=["vg11","@node"] } where "node" is the value of "uname -n" command
  Using this configuration instructs the agent to not use tagging while activating the volume group vg11. The rest of the volume groups require tagging however.

Sample configurations for LVMVolumeGroup agent

The sample configurations for this agent follows:

Linux configuration 1

In this example, the volume group testvg_1 is created on disks that have multiple paths and have LVM tagging enabled.

```
LVMVolumeGroup lvg1 (
VolumeGroup = testvg_1
EnableLVMTagging = 1
)
```

Linux Configuration 2

In this example, the volume groups testvg_1 and testvg_2 are created on disks that have multiple paths. LVM tagging is enabled for testvg_1 and disabled for test_vg2.

```
LVMVolumeGroup lvg1 (
VolumeGroup = testvg_1
EnableLVMTagging = 1
)
LVMVolumeGroup lvg2 (
VolumeGroup = testvg_2
)
Here the /etc/lvm/lvm_`uname -n`.conf file contains:
activation { volume_list=["testvg_2","@node"] }
where node is the value of uname -n command.
```

Debug log levels for LVMVolumeGroup agent

The LVMVolumeGroup agent uses the following debug log levels:

DBG_1, DBG_3, DBG_5

Mount agent

The Mount agent brings online, takes offline, and monitors a file system or an NFS client mount point. You can use the agent to make file systems or NFS client mount points highly available.

This agent also supports high availability fire drills.

For important information about this agent, See "Notes for Mount agent" on page 71.

IMF awareness

The Mount agent is IMF-aware and uses Asynchronous Monitoring Framework (AMF) kernel driver for IMF notification. For more information about IMF and intelligent resource monitoring, refer to the *Cluster Server Administrator's Guide*.

Note: IMF for mounts is supported only for VxFS, ext4, and XFS file system types.

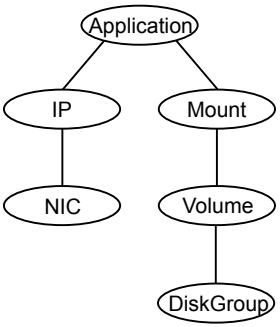
For more information about IMF-related Mount agent functions, see Agent functions for Mount agent.

For more information about using the Mount agent with IMF, see IMF usage notes.

Dependencies for Mount agent

The Mount resource does not depend on any other resources.

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



Agent functions for Mount agent

Online	Mounts a block device on the directory. If the mount process fails for non-NFS mounts, the agent attempts to run the fsck command on the device before attempting to mount the file system again. If file system type is NFS, agent mounts the remote file system to a specified directory. The remote NFS file system is specified in the BlockDevice attribute. **Note:** If you enable writeback mode for the SmartIO feature, the files that have pending dirty data can become inaccessible in the event of a node failure or Solid State Drive (SSD) device failure. For more information about recovering the writeback cache, refer to the *Veritas InfoScale SmartIO for Solid State Drives Solutions Guide*.
Offline	Unmounts the mounted file system.

Monitor	Determines if the file system is mounted. If IMF is enabled for the Mount agent, the resource is monitored asynchronously and any change in the resource state is immediately sent to VCS for appropriate action.
imf_init	Initializes the agent to interface with the asynchronous monitoring framework (AMF) kernel driver. This function runs when the agent starts up.
imf_getnotification	Waits for notification about resource state changes. This function runs after the agent initializes with the AMF kernel driver. The agent continuously waits for notification and takes action on the resource upon notification.
imf_register	Registers the resource entities, which the agent must monitor, with the AMF kernel driver. This function runs for each resource after the resource goes into steady state (online or offline). This action entry point registers the mount point, block device, and file system type for mount agent.
Clean	Unmounts the mounted file system forcefully.

Info

The Mount agent info function executes the command:

```
# df -h mount_point
```

The output displays Mount resource information:

```
Size Used Avail Use%
```

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

```
# haconf -makerw

# hatype -modify Mount InfoInterval 60
```

The command to retrieve information about the Mount resource is:

```
# hares -value mountres ResourceInfo
```

Output includes:

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

Action	■ chgmntlock Resets the VxFS file system lock to a VCS-defined lock. ■ mountpoint.vfd Checks if the specified mount point exists on the offline node. If it fails and you request that VCS fixes it, it creates the mount point directory using `mkdir` command. ■ mounted.vfd Checks if the mount point is already mounted on the offline node. If it fails, you need to unmount all the file systems from the specified mount point directory. ■ vxfslic.vfd Checks for valid Veritas File System (VxFS) licenses. If it fails, you need to update the license for VxFS. ■ mountentry.vfd Checks that the mount point is not listed in auto file system tables. For example, /etc/fstab If this action fails, you need to remove the mount point from auto file system tables.
attr_changed	Unlocks the mounts when you change the value of the VxFSMountLock attribute from 1 to 0.

State definitions for Mount agent

The state definitions for this agent follow:

ONLINE	For the local file system, indicates that the block device is mounted on the specified mount point. For an NFS client, indicates that the NFS remote file system is mounted on the specified mount directory.
OFFLINE	For the local file system, indicates that the block device is not mounted on the specified mount point. For an NFS client, indicates that the NFS remote file system is not mounted on the specified mount directory.

FAULTED	For the local file system, indicates that the block device has unexpectedly unmounted. For the NFS client, indicates that the NFS remote file system has unexpectedly unmounted.
UNKNOWN	Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

Attributes for Mount agent

Table 2-13 Required attributes

Required attribute	Description

BlockDevice	Block device for mount point.
	For LVM2, use the actual mapper path to the volume.
	Type and dimension: string-scalar
	Examples:
	■ LVM2 example
	`# ls -la /dev/ora_vg/ora_vol`
	`lrwxrwxrwx  1 root root 26 Apr 17 04:48`
	`/dev/ora_vg/ora_vol -> /dev/mapper/ora_vg-`
	`ora_vol`
	Use the path /dev/mapper/ora_vg-ora_vol for the BlockDevice attribute.
	■ NFS example
	If the file system type is NFS, then specify the BlockDevice as: server:/path/to/share
	IPv4
	sys1.example.com:/usr/share1
	IPv6
	[2001::5]:/path/to/share
	■ "/dev/vx/dsk/myvcs_dg/myvol"

Table 2-13 Required attributes *(continued)*

Required attribute	Description
FsckOpt	Mandatory for non-NFS mounts. Use this attribute to specify options for the `fsck` command. You must correctly set this attribute for local mounts. If the mount process fails, the `fsck` command is executed with the specified options before it attempts to remount the block device. Its value must include either `-y` or `-n`. Refer to the fsck manual page for more information. The `-y` argument enables the VxFS file systems to perform a log replay before a full `fsck` operation. For NFS mounts, the value of this attribute is not applicable and is ignored. Type and dimension: string-scalar VxFS example: -y **Note:** When you use the command line, add the % sign to escape '-'. For example: hares -modify MntRes FsckOpt %-y
FSType	Type of file system. Supports vxfs, bind, ext2, ext3, ext4, xfs, nfs, or reiserfs. Type and dimension: string-scalar
MountPoint	Directory for mount point. Type and dimension: string-scalar Example: "/mnt/apache1"

Table 2-14 Optional attributes

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

Table 2-14 Optional attributes *(continued)*

Optional attribute	Description
MountOpt	Options for the `mount` command. Refer to the *Mount* manual page for more information. Do not specify `-o` in the MountOpt field. The agent uses this option only when bringing a Mount resource online. Type and dimension: string-scalar Example: "rw"
VxFSMountLock	This attribute is applicable to Veritas File System (VxFS). It controls the agent's use of the locking feature provided by vxfs to prevent accident unmounts. If the value of this attribute is 0, the agent does not lock the mount point when the resource is brought online. It does not monitor the status of the lock when the resource is online. No warnings appear if the mount has been locked with a key different than "VCS". If the value of this attribute is 1, during online, the agent uses the key "VCS" to lock the mount point. The monitor agent function monitors the locks during every cycle. ■ If the mount point is not locked, the agent locks it. ■ If the mount point is already locked with a key other than "VCS", the agent logs a warning. It then requests that you run the Chgmntlock action agent function. During offline, the agent, as required, unlocks using whatever key needed. Type and dimension: boolean-scalar Default: 1
SnapUmount	If the value of this attribute is 1, this attribute automatically unmounts VxFS snapshots when the file system is unmounted. If the value of this attribute is 0, and snapshots are mounted, the resource cannot be brought offline. In this case, failover does not occur. Type and dimension: boolean-scalar Default: 0

Table 2-14 Optional attributes *(continued)*

Optional attribute	Description
AccessPermissionChk	If the value of this attribute is set to 1 or 2, the monitor verifies that the values of the MntPtPermission, MntPtOwner, and MntPtGroup attributes are the same as the actual mounted file system values. If any of these do not match the values that you have defined, a message is logged. If the value of this attribute is 2, and if the mounted file system permissions do not match the attribute values, the Monitor agent function returns the state as OFFLINE. Type and dimension: integer-scalar Default: 0
CreateMntPt	If the value of this attribute is 0, no mount point is created. The mount can fail if the mount point does not exist with suitable permissions. If the value of this attribute is 1 or 2, and a mount point does not exist, the agent creates a mount point with system default permissions when the resource is brought online. If the permissions of the mount point is less than 555, a warning message is logged. If the value of this attribute is 2, and the mount point does not exist, the agent creates a mount point with system default permissions when the resource is brought online. If the permissions for the mount point are less than 555, a warning message is logged. In addition, VCS deletes the mount point and any recursively created directories when the resource is brought offline. The mount point gets deleted only if it is empty, which is also true for recursive mount points. Type and dimension: integer-scalar Default: 0
MntPtGroup	This attribute specifies the group ownership of the mounted file system. The agent verifies the group ownership of the mounted file system every monitor cycle if the value of the AccessPermissionChk attribute is not 0. Type and dimension: string-scalar Example: "grp1"

Table 2-14 Optional attributes *(continued)*

Optional attribute	Description
MntPtOwner	This attribute specifies the user ownership of the mounted file system. The agent verifies the user ownership of the mounted file system every monitor cycle if the value of the AccessPermissionChk attribute is not 0. Type and dimension: string-scalar Example: "usr1"
MntPtPermission	This attribute specifies the permissions of the mounted file system in absolute format of a four-digit octal. The agent verifies the mode of the mounted file system every monitor cycle if the value of the AccessPermissionChk attribute is not 0. Type and dimension: string-scalar Example: "0755"
OptCheck	The value of this attribute determines if VCS should verify the mount options. The state of the resource is determined based on the result of the verification. If set to 0 (default), the mount options are not checked. If the value of the OptCheck attribute is 1, 2 or 3, a check is performed to see if the mount command options that you have specified for VCS are set in the MountOpt attribute. The MountOpt attributes should be the same as the actual mount command options. If the actual mount options differ from the MountOpt attribute, a message is logged. The state of the resource depends on the value of this attribute. If the value of the attribute is 1, the state of the resource is unaffected. If the value is 2, the state of the resource is set to offline. If the value is 3, state of the resource is set to unknown. Type and dimension: integer-scalar Default: 0
RecursiveMnt	If the value of this attribute is 1, VCS creates all the parent directories of the mount point if necessary. Type and dimension: boolean-scalar Default: 0

Table 2-14 Optional attributes *(continued)*

Optional attribute	Description
ReuseMntPt	If the same mount point needs to be specified in more than one mount resource, set the value of this attribute to 1. Note that this attribute only accepts a value of 1 or 0. To use this attribute, the cluster administrator needs to add this attribute to the arglist of the agent. Set the appropriate group and resource dependencies such that only one resource can come online on a system at a time. Type and dimension: integer-scalar Default: 0
CacheRestoreAccess	This attribute is applicable only if: ■ File system type is VxFS. ■ Writeback caching is enabled for the SmartIO feature. The value of this attribute determines whether to perform restore access operation or not. The following are the values: ■ 0: Does not perform restore access operation. ■ 1: Performs restore access operation. Type and dimension: boolean-scalar Default: 0 **Note:** For the procedure and implications of enabling writeback caching, refer to the *Veritas InfoScale SmartIO for Solid State Drives Solutions Guide*.

Table 2-15 Internal attribute

Internal attribute	Description
AEPTimeout	This is an internal attribute. Do not modify this attribute. This attribute is used to pass the timeout value of agent entry points.

Resource type definition for Mount agent

The resource definition for this agent on Linux follows:

```
type Mount (
        static keylist RegList = { VxFSMountLock }
```

```
        static int IMF{} = { Mode = 3, MonitorFreq = 1,
         RegisterRetryLimit = 3 }
        static str IMFRegList[] = { MountPoint, BlockDevice, FSType }
        static boolean AEPTimeout = 1
        static keylist SupportedActions = { "mountpoint.vfd",
         "mounted.vfd", "vxfslic.vfd" , "chgmntlock", "mountentry.vfd" }
        static str ArgList[] = { MountPoint, BlockDevice, FSType,
         MountOpt, FsckOpt, SnapUmount, CkptUmount, OptCheck,
         CreateMntPt, MntPtPermission, MntPtOwner, MntPtGroup,
         AccessPermissionChk, RecursiveMnt, VxFSMountLock,
         CacheRestoreAccess }
        str MountPoint
        str BlockDevice
        str FSType
        str MountOpt
        str FsckOpt
        boolean SnapUmount = 0
        boolean CkptUmount = 1
        int OptCheck = 0
        int CreateMntPt = 0
        int ReuseMntPt = 0
        str MntPtPermission
        str MntPtOwner
        str MntPtGroup
        int AccessPermissionChk = 0
        boolean RecursiveMnt = 0
        boolean VxFSMountLock = 1
        boolean CacheRestoreAccess = 0
)
```

Notes for Mount agent

The Mount agent has the following notes:

- Support for spaces in directory names
- Support for multiple bindfs
- High availability fire drill
- VxFS file system lock
- IMF usage notes
- Enabling Level two monitoring for the Mount agent
- RHEL 7: NFS file system version

- RHEL 7: Configuring bind mounts
- Support for Amazon EFS

Support for spaces in directory names

The Mount agent supports directory names with spaces. The space can be leading, trailing, or in the middle of the name. If the directory name has a trailing space, provide an extra "/" at the end of the corresponding attribute of a Mount resource. The attributes that currently support spaces in directory names are MountPoint and BlockDevice. Note that the agent does not support spaces created using the TAB key.

Support for multiple bindfs

The Mount agent supports file system of the type bind. It also allows multiple mounts of type bind from the same block device. However, the mount points are different. So, with the FSType attribute set to bind, the mount resource does not report UNKNOWN state when the same block device is mounted on another mount point.

Sample configuration

```
Mount bindmount (
         MountPoint = "/bind"
         BlockDevice = "/test/bind"
         FSType = bind
         MountOpt = rw
         )

Mount bindmount1 (
         MountPoint = "/bind1"
         BlockDevice = "/test/bind"
         FSType = bind
         MountOpt = rw
         )

Mount mount (
         MountPoint = "/test"
         BlockDevice = "/dev/vx/dsk/testdg/testvol"
         FSType = vxfs
         MountOpt = rw
         FsckOpt = "-y"
```

```
        )

    bindmount requires mount
    bindmount1 requires mount
```

High availability fire drill

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

For Mount resources, the high availability drill performs the following, it:

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

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

VxFS file system lock

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

```
# mount
```

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

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

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

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

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

IMF usage notes

If you use IMF for intelligent resource monitoring, depending on the value of the FSType attribute, you must set the MonitorFreq key value of the IMF attribute.

If the FSType attribute value is bind, IMF registration on Linux for “bind” file system type is not supported.

IMF should not be enabled for the resources where the BlockDevice can get mounted on multiple mount points. If the FSType attribute value is nfs, then IMF registration for “nfs” file system type is not supported.

See the *Cluster Server Administrator’s Guide* for the IMF attribute description.

Enabling Level two monitoring for the Mount agent

Level two monitoring can be enabled for the Mount agent only if FSType is set to "nfs".

To enable Level two monitoring, run the following commands:

- `# haconf -makerw`
- `# hares -override resource_name LevelTwoMonitorFreq`
- `# hares -modify resource_name LevelTwoMonitorFreq 1`
- `# haconf -dump -makero`

For more details about the LevelTwoMonitorFreq attribute, refer to the *Cluster Server Agent Developer’s Guide*.

RHEL 7: NFS file system version

On RHEL 7, the `mount` command displays the file system type as either `nfs` or `nfs4` depending on the NFS version used in the NFS server. If the NFS server is using NFSv4, the file system type is displayed as `nfs4`, so you must set `FSType = nfs4` for the mount resource. If the NFS server is using NFSv3, the file system type is displayed as `nfs`, so you must set `FSType = nfs` for the mount resource.

You can override the NFS version setting by specifying `vers` when you mount the file system.

The following is an example in which the default NFS version is used:

```
# mount -t nfs nfsserver:/mnt /mnt

# mount

...
nfsserver:/mnt on /mnt type nfs4(rw,relatime,vers=4.0,rsize=1048576,
```

```
wsize=1048576,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,
sec=sys,clientaddr=192.168.0.10,local_lock=none,addr=192.168.0.10)
#
```

The following is an example in which `vers=3` is specified:

```
# mount -t nfs -o vers=3 nfsserver:/mnt /mnt

# mount

...
nfsserver:/mnt on /mnt type nfs(rw,relatime,vers=3,rsize=1048576,
wsize=1048576,namlen=255,hard,proto=tcp,timeo=600,retrans=2,
sec=sys,mountaddr=192.168.0.10,mountvers=3,mountport=44649,
mountproto=udp,local_lock=none,addr=192.168.0.10)
#
```

RHEL 7: Configuring bind mounts

RHEL 7 supports bind mount, but it does not display the bind flag with the mount point. In case of bind mounts, the `mount` command displays the parent block device and file system type instead of the block device and "bind" option used while performing a bind mount. The `mount` command output displays the bind mount as a parent block device mounted on multiple mount points. Due to this behavior, the Mount agent is unable to determine bind mounts and any mount resource with `FSType=bind` fails to come online and the Mount agent reports the resource state as UNKNOWN. You can configure `FSType=bind` in other Linux versions.

The Mount agent now allows the same block device to be mounted on multiple mount points, so to configure a bind mount in VCS, use the parent block device and file system type.

RHEL 6: Sample bind mount configuration

```
Mount mntres1 (
                BlockDevice = "/dev/vx/dsk/vcsdg/vcsvol"
                MountPoint = "/vcsmnt"
                FSType = vxfs
                FsckOpt = "-y"
                )

Mount mntres2 (
                BlockDevice = "/vcsmnt"
                MountPoint = "/bindmount"
                FSType = bind
                )
```

RHEL 7: Sample bind mount configuration

Set `VxFSMountLock = 1` only for the topmost resource in the resource dependency tree. For all the other mount resources, set `VxFSMountLock = 0`. For bind mount, the kernel simply clones the mount options and hence the pseudo file inherits the same mount lock information locking the other resources.

```
Mount share_mnt2_bind (
                MountPoint = "/bind_point2"
                BlockDevice = "/dev/vx/dsk/bind_dg/bind_vol"
                FSType = vxfs
                FsckOpt = "-y"
               VxFSMountLock = 1
                )

Mount share_mnt1_bind (
                MountPoint = "/bind_point"
                BlockDevice = "/dev/vx/dsk/bind_dg/bind_vol"
                FSType = vxfs
                FsckOpt = "-y"
                VxFSMountLock = 0
                )


 Mount share_mnt_bind (
                MountPoint = "/mount_point"
                BlockDevice = "/dev/vx/dsk/bind_dg/bind_vol"
                FSType = vxfs
                FsckOpt = "-y"
                VxFSMountLock = 0
                )
        // resource dependency
        share_mnt2_bind requires share_mnt1_bind
        share_mnt1_bind requires share_mnt_bind
        share_mnt_bind requires dg_res_bind
```

Support for Amazon EFS

The Mount agent supports Amazon Elastic File System (Amazon EFS). In case of multi-node clusters, the agent mounts the file system on the appropriate nodes in parallel.

In case of EFS, the BlockDevice attribute of this agent is set to the Amazon EFS endpoint. For example:

```
BlockDevice = "fs-1a2b3c4d.efs.us-east-1.amazonaws.com:/"
```

Sample configurations for Mount agent

Basic configuration for Mount agent

Configuration for Linux follows:

```
Mount mnt_r1_1 (
MountPoint = "/testdir/LVM_R1_1"
BlockDevice = "/dev/mapper/emc_vg-emc_r1"
FSType = ext3
MountOpt = "rw"
FsckOpt = "-y"
)
```

VxFS mount lock example for Mount agent

```
Mount test_mnt (
MountPoint = "/home/export"
BlockDevice = "/dev/vx/dsk/nfsdg/vol0"
FSType = vxfs
MountOpt = rw
FsckOpt = "-n"
VxFSMountLock = 1
)
```

NFS mount example for Mount agent

```
Mount mnt1 (
MountPoint = "/mnt/vcstest"
BlockDevice = "sys1:/home/export"
FSType = nfs
)
```

EFS mount example for Mount agent

```
Mount mountefs_1 (
  MountPoint = "/databaseefs"
  BlockDevice = "fs-1a2b3c4d.efs.us-east-1.amazonaws.com:/"
  FSType = nfs4
  FsckOpt = "-n"
)
```

Debug log levels for Mount agent

The Mount agent uses the following debug log levels:

DBG_1, DBG_2, DBG_3, DBG_4, DBG_5

VMwareDisks agent

The VMwareDisks agent enables vMotion and VMware Distributed Resource Scheduler (DRS) in VCS clusters configured and deployed on virtual machines in VMware environment.

When a VCS cluster with a shared disk is configured on virtual machines, VMware does not support VMware Distributed Resource Scheduler (DRS) and vMotion. Thus the vMotion and DRS capabilities are compromised. The solution to this issue is to attach the disk(s) to a single virtual machine at a time in a VCS cluster. In case of a user-initiated failover or a fault-induced failover, these disks failover (detach-attach) to the target virtual machine along with the service group. VMwareDisks agent manages the attaching and detaching of the disks to the virtual machines.

To ensure proper functioning of the VMwareDisks agent, verify the following:

- Ensure that the VMware disks are in persistent mode. If the VMware disks are in independent mode, VMWareDisks agent will revert them to persistent mode in case of a failover.
- The ESX/ESXi host or vCenter user account has administrative privileges or is a root user. If you do not want to use the administrator user account or the root user, create a role with the required privileges for the VMwareDisks resource functionality and assign this role to one or more users. This user must have the ability to:
  - Perform low-level file operations
  - Add an existing disk
  - Modify resources
  - Remove disks

  Note that the above list is only illustrative, you can add additional privileges as required.

  If the user is a vCenter user, you must assign the requisite privileges so that the user can access the datastore.

To assign role and privileges

1. Log on to the vCenter Server and navigate to **Home > Inventory > Datastores and Datastore Clusters**.
2. From the vCenter inventory tree view, select the appropriate datacenter.
3. Right-click the datacenter and select **Add Permission**.

   Or

   Go to the **Permissions** tab. On the Permissions pane, right-click and select **Add Permission**.
4. In the Assign Permissions window, add the user, select the role, and assign privileges.

   For more information, refer to *VMware vSphere ESXi and vCenter Server Documentation*.

Agent functions

Online	Attaches the disks to the virtual machine.
Offline	Detaches the disks from the virtual machine.
Monitor	Verifies that the disks are attached to the virtual machine.

State definitions

ONLINE	Indicates that the disks are attached to the virtual machine.
OFFLINE	Indicates that the disks are not attached to the virtual machine.
UNKNOWN	Indicates that the agent could not detect the state of the disks.

Attributes

This section summarizes the required and optional attributes of the VMwareDisks agent.

Table 2-16 Required attributes

Required attribute	Description
ESXDetails	The list of hostnames and credentials of the ESX hosts or vCenter on which the virtual machines are configured. The key is ESX or vCenter hostname or IP address. The value must be in the format: ■ ESX or IP address—'User name'='Encrypted password' ■ vCenter hostname—'Domain\User Name'='Encrypted password' Type and dimension: string-association
DiskPaths	The list of disks paths to be managed by the VMwareDisks resource. The key is of the form '[Data store name] Disk path'. The value is of the form 'SCSI controller key: Target ID (unit number of the disk)'. For example, 0:2. For RDM, add prefix 'RDM:' to the disk path. Alternatively, the key can be of the form 'Disk_UUID: [Data store name] Disk_path'. In case of RDM, 'RDM:Disk_UUID:[Data store name] Disk_path'. If the disk UUID is not provided, the agent will discover the UUID and modify the attribute to have UUID of the disk. Type and dimension: string-association

Table 2-17 Optional attributes

Optional attribute	Description
HAInfoDetails	Determines whether or not vSphere HA is enabled. This attribute uses the vCenter Server hostname or IP address to determine the status of vSphere HA. The value must be specified in the format: Key=Value. Where: ■ Key= vCenter Server hostname or IP address ■ Value=vCenter Server logon user credentials. This must be specified in the format: 'Domain\User Name'='Encrypted password' If you do not specify a value for this attribute, the agent considers the vSphere HA setting based on the IsVMHAEnabled attribute value. Type and dimension: string-association

Table 2-17 Optional attributes *(continued)*

Optional attribute	Description
IsVMHAEnabled	Defines whether vSphere HA is enabled. The value 1 indicates that vSphere HA is enabled. This attribute value should match the vSphere HA settings in the VMware vSphere HA cluster. If vSphere HA is turned on, set this attribute value to 1 (True). If vSphere HA is turned off, set this attribute value to 0 (False). vSphere HA settings are considered based on this attribute only if the HAInfoDetails attribute is not set, or if the agent fails to retrieve the vSphere HA details based on the HAInfoDetails attribute. Default: 1 (True) Type and dimension: boolean-scalar
PanicVMOnESXLoss	Set this attribute value to 1 (True) to trigger panic on the virtual machine when the ESX host loses network connectivity. Default: 0 (False) Type and dimension: boolean-scalar
ForceRegister	For internal use only.
VMRegisterWait	The specified time interval, in seconds, during which VMware HA registers the virtual machine on any other ESX host when an ESX host fails. This is applicable only if IsVMHAEnabled is set to true. Default value: 120 seconds Type and dimension: integer
VirtualDiskMode	Specifies the mode to be used when the disk is attached. You can set the value of this attribute to one of the following: ■ `persistent` ■ `independent_persistent` ■ `independent_nonpersistent` You must modify the value after you configure application monitoring. **Note:** The VMwareDisks agent does not detect the mode in which the disk is configured. After a failover, the disk is attached in the mode that is defined in the attribute value. For details about the disk modes, refer to the VMware documentation. Default value: `persistent` Type and dimension: string-association

Resource type definition

```
type VMwareDisks (
 static keylist RegList = { ESXDetails }
 static keylist SupportedActions = { checkESXconn }
 static int InfoInterval = 60
 static int NumThreads = 1
 static str ArgList[] = { ESXDetails, DiskPaths, ResourceInfo,
  IsVMHAEnabled, VMRegisterWait, VirtualDiskMode, PanicVMOnESXLoss,
  ForceRegister, HAInfoDetails }
 str ESXDetails{}
 str DiskPaths{}
 str HAInfoDetails{}
 boolean IsVMHAEnabled = 1
 int VMRegisterWait = 120
 str VirtualDiskMode = persistent
 boolean PanicVMOnESXLoss = 0
 temp boolean ForceRegister = 0
)
```

Sample configurations

Sample configuration where UUID is not provided for the DiskPaths attribute:

```
VMwareDisks VMwareDisks_1 (
 ESXDetails = { "192.168.0.100" = "root=HVJtWTwVLnINjNK",
  "192.168.0.101" = "root=HVJtWTwVLnINjNK",
  "192.168.0.102" = "root=HVJtWTwVLnINjNK" }
 DiskPaths = {
  "[SharedStorage2] VxSwapHost2_1/VxSwapHost1_1.vmdk" = "0:1",
  "[SharedStorage2] VxSwapHost2_1/VxSwapHost1_2.vmdk" = "0:2",
  "RDM:[SharedStorage2] VxSwapHost2_1/VxSwapHost1_3.vmdk" = "0:3" }
 VirtualDiskMode = independent_persistent
)
```

Sample configuration for vCenter:

```
VMwareDisks VMwareDisks_1 (
ESXDetails = { "192.168.0.100" = "administrator=HVJtWTwVLnINjNK" }
DiskPaths = {
 "[SharedStorage2] VxSwapHost2_1/VxSwapHost1_1.vmdk" = "0:1",
 "[SharedStorage2] VxSwapHost2_1/VxSwapHost1_2.vmdk" = "0:2",
 "RDM:[SharedStorage2] VxSwapHost2_1/VxSwapHost1_3.vmdk" = "0:3" }
```

```
VirtualDiskMode = independent_persistent
)
```

Sample configuration where UUID is provided for the DiskPaths attribute:

```
VMwareDisks VMwareDisks_1 (
ESXDetails = { "192.168.0.100" = "root=HVJtWTwVLnINjNK",
 "192.168.0.101" = "root=HVJtWTwVLnINjNK",
 "192.168.0.102" = "root=HVJtWTwVLnINjNK" }
DiskPaths = {"RDM:6000C29a-11a3-7845-029d-10737a83ced7:
 [SharedStorage2] VxSwapHost2_1/VxSwapHost1_3.vmdk" = "0:3" }
VirtualDiskMode = independent_persistent
)
```

SFCache agent

The SmartIO feature of Veritas InfoScale enables data efficiency on your SSDs through I/O caching. Using SmartIO to improve efficiency, you can optimize the cost per I/O per second (IOPS). SmartIO uses advanced, customizable heuristics to determine what data to cache and how that data gets removed from the cache. The heuristics take advantage of Veritas InfoScale's knowledge of the characteristics of the workload.

SmartIO uses a cache area on the target device or devices. The cache area is the storage space that SmartIO uses to store the cached data and the metadata about the cached data. The type of the cache area determines whether it supports VxFS caching or VxVM caching. To start using SmartIO, you can create a cache area with a single command, while the application is online.

For more information about SmartIO, see *Veritas InfoScale SmartIO for Solid State Drives Solutions Guide*.

The SFCache agent enables, disables, and monitors cache. In case of a cache fault, the application still runs without any issues on the very same system, but with degraded I/O performance. Considering this, the SFCache agent provides an attribute to control the agent behavior. You can either choose to "IGNORE" or initiate “FAILOVER” in case of cache fault.

The SmartIO feature allows more than one cache area for VxFS and one cache area for VxVM on a single node; all object-level caches are created in these cache areas. An SFCache resource is configured per object (either mount point or volume) for which the SmartIO feature needs to be enabled. For VxFS caching, the SFCache resource depends on the Mount or CFSMount resource. For VxVM caching, the SFCache resource depends on the DiskGroup, Volume, VolumeSet, or CVMVolDg resource.

If the SmartIO feature is not enabled on a node, the SFCache agent will work as a simple FileOnOff agent. The SFCache resource state will be reported as ONLINE/OFFLINE, but caching-related operations will not be performed.

Resource dependency

Figure 2-10 shows sample SFCache resource dependency for VxFS caching. The SFCache resource can depend on the Mount or CFSMount resource.

Figure 2-10 Sample SFCache resource dependency (VxFS caching)

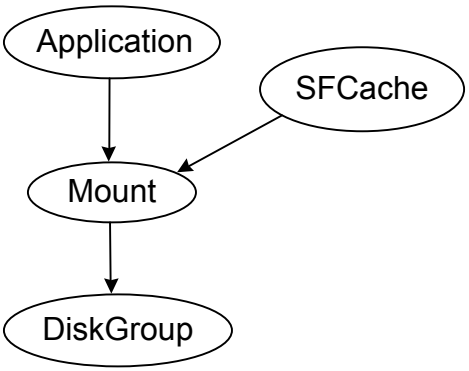
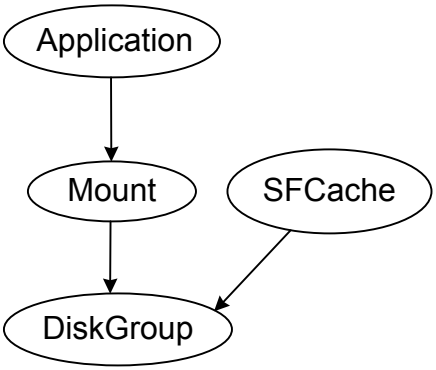


Figure 2-11 shows sample SFCache resource dependency for VxVM caching. The SFCache resource can depend on the DiskGroup, Volume, VolumeSet, or CVMVolDg resource.

Figure 2-11 Sample SFCache resource dependency (VxVM caching)



Agent functions

Online	Sets the cache mode as configured in the CacheMode attribute and finds out the type of cache area. Following are the types: ■ VxFS ■ VxVM
Offline	Disables the cache for a configure object.
Clean	Disable the cache for a configured object.
Monitor	Monitors the state of the caching for a configured cache object.
Attr_changed	Validates and makes necessary changes if the CacheMode attribute is reconfigured.

State definitions

The state definitions for this agent follow:

ONLINE	Indicates that caching is enabled for the configured cache object using the appropriate mode. If caching cannot be done on a node, the resource reports ONLINE even if there is no caching.
OFFLINE	Indicates that caching is disabled or the configured cache object is not active.
FAULTED	Indicates that caching is disabled for the specified cache object.
UNKNOWN	Indicates that invalid caching mode is specified for a cache object or the specified resource configuration is invalid.

Attributes

This section describes the attributes of the SFCache agent.

Table 2-18 Required attributes

Attribute	Description
CacheObjectName	Specifies the cache object name; it can be a mount point or disk group/volume. Type and dimension: string-scalar

Table 2-18 Required attributes *(continued)*

Attribute	Description

CacheArea	Specifies the name of the cache areas.
	The writeback cache area is valid only when `CacheMode = writeback` and if it is a VxFS cache area.
	Type and dimension: string-association
	Example:
	`CacheArea @sys1 = { Read = sfcachearea_11, Writeback = sfcachearea_12 }`
	`CacheArea @sys2 = { Read = sfcachearea_21, Writeback = sfcachearea_22 }`

Table 2-19 Optional attributes

Attribute	Description
CacheMode	Specifies the caching mode. Following are the caching modes: ■ read ■ writeback ■ nocache Type and dimension: string-scalar Default: read
CacheFaultPolicy	Specifies the action to be performed in case of a cache fault. Following are the values: ■ IGNORE ■ FAILOVER Type and dimension: string-scalar Default: IGNORE

Table 2-20 Internal attributes

Attribute	Description
FaultOnMonitorTimeouts	Defines whether VCS interprets the Monitor timeout as a resource fault. By default, the FaultOnMonitorTimeouts attribute is set to 4, but the SFCache agent overrides this value and sets it to 0. The Monitor function must timeout four times in a row before the resource is marked as faulted. The first monitor timeout timer and the counter of timeouts are reset after one hour of the first monitor timeout. If the attribute is set to 0, VCS does not treat Monitor timeout as a resource fault. If the attribute is set to 1, VCS interprets the Monitor timeout as a resource fault and the agent calls the Clean function to shutdown the resource. **Note:** If the CacheFaultPolicy attribute is set to FAILOVER, Veritas recommends to set the FaultOnMonitorTimeouts attribute to 4. Default: 0
NumThreads	Number of threads that are used within the agent process for managing resources. This number does not include the number of threads that are used for other internal purposes. Setting the NumThreads attribute to a higher value may decrease the time required to go online or the time required to monitor a large number of SFCache resources. **Note:** If the NumThreads value is greater than 1, then there is a possibility that the `sfcache` command may fail. Type and dimension: static integer-scalar Default: 1

Resource type definition

```
type SFCache (
        static boolean IntentionalOffline = 1
        static int NumThreads = 1
        static int FaultOnMonitorTimeouts = 0
        static keylist RegList = { CacheMode }
        static str ArgList[] = { CacheObjectName,
         CacheArea, CacheMode, CacheFaultPolicy }
        str CacheObjectName
        str CacheArea{} = { Read=NONE, Writeback=NONE }
```

```
        str CacheMode = read
        str CacheFaultPolicy = IGNORE
)
```

Notes for SFCache agent

The SFCache agent has the following note:

- Configuring SFCache resource with CVM/CFS

Configuring SFCache resource with CVM/CFS

To use the SFCache agent with CFS setup, you must configure the SFCache resource in a separate parallel service group with online local soft dependency between the SFCache service group and CFSMount service group.

The following is a sample configuration:

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

        CFSMount cfsmount1 (
                MountPoint = "/cfsmnt"
                BlockDevice = "/dev/vx/dsk/cfsdg01/cfsvol01"
                MountOpt @sysA = rw
                MountOpt @sysB = rw
                NodeList = { sysA, sysB }
                )

        CVMVolDg cvmvoldg1 (
                CVMDiskGroup = cfsdg01
                CVMVolume = { cfsvol01 }
                CVMActivation @sysA = sw
                CVMActivation @sysB = sw
                )

        requires group cvm online local firm
        cfsmount1 requires cvmvoldg1

group cfs_sfcache_sg (
        SystemList = { sysA = 0, sysB = 1 }
        Parallel = 1
```

```
AutoStartList = { sysA, sysB }
)

SFCache sfcache1 (
        CacheMode = read
        CacheFaultPolicy = IGNORE
        CacheObjectName = "/cfsmnt"
        CacheArea @sysA = { Read = sfcachearea_11 }
        CacheArea @sysB = { Read = sfcachearea_21 }
        )

requires group cfssg online local soft
```

Debug log levels

The SFCache agent uses the following debug log levels:

DBG_3, DBG_4

AWS EBSVol agent

A dedicated agent is required to provide high availability of the Amazon EBS volumes across nodes in a VCS cluster within the same availability zone. The EBSVol agent provides high availability of the Amazon EBS volumes during fail-over of an application.

The EBSVol agent monitors the Amazon EBS volumes. It also attaches or detaches the Amazon EBS volumes to and from the EC2 instances respectively. This agent uses AWS CLI commands to determine the state of the Amazon EBS volumes.

Note: The Amazon EBS volumes should not be configured in a parallel service group.

Prerequisites

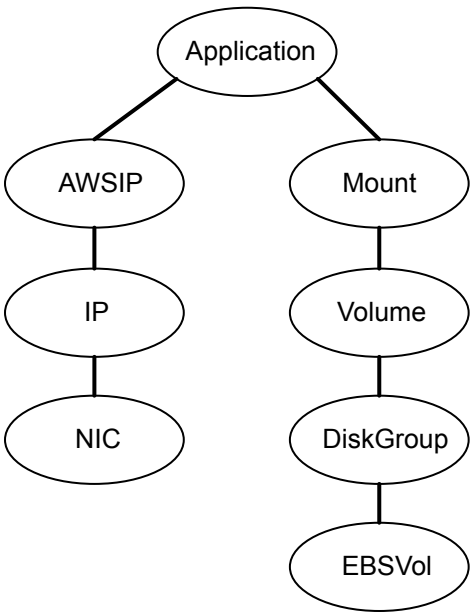
- Install the AWS CLI package.
- Create an IAM role with requisite permissions and attach the roles to the EC2 cluster instances.
  Go to **IAM** > **Create Policy** > **Role** and create the role.
  The following is a sample of the policy .json file:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                 "ec2:DescribeVolumes",
                 "ec2:AttachVolume",
                 "ec2:DetachVolume"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
```

Dependencies

The EBS volume resource does not depend on any other resources.

Figure 2-12 Sample service group that includes a EBS volume resource



Agent functions

Online	Attaches the Amazon EBS volume to an EC2 instance using the attach-volume command.
Offline	Detaches the Amazon EBS volume from the EC2 instance using the detach-volume command.
Monitor	Determines if the Amazon EBS volume is attached to the EC2 instance or not using the `describe-volume` command.
Clean	Adds the following warning in the agent log if the EBS Volume is busy or cannot be detached. `Administrative intervention required.` Administrator should take corrective actions to detach the EBS Volume.

State definitions

ONLINE	Indicates that the Amazon EBS volume is attached to an EC2 instance or is in busy state.
OFFLINE	Indicates that: ■ EBS Volume attachment is in progress ■ EBS Volume detachment is in progress ■ EBS Volume is not attached to any EC2 instance ■ EBS Volume is attached to a different EC2 instance ■ EBS Volume is attached to the same EC2 instance, but as a different device
FAULTED	Indicates that the EBS volume has unexpectedly got detached from an EC2 instance.
UNKNOWN	Indicates that a problem exists because of one of the following reasons: ■ AWS CLI is not installed. ■ AWSBinDir attribute is not configured correctly. For example, if AWS CLI is installed in `/usr/local/bin`, the AWSBinDir attribute should be set to `/usr/local/bin`. ■ Roles are not configured correctly or attached to the instance.

Attributes

Table 2-21 Required attributes

Attribute	Description
VolumeId	The ID of the Amazon EBS volume. The volume and the EC2 instance must be within the same availability zone. Type and dimension: string-scalar

Table 2-22 Optional attributes

Attribute	Description
AWSBinDir	Location of the AWS EC2 commands and binaries. Type and dimension: string-scalar

Table 2-23 Internal attributes

Attribute	Description
Device	Name of the device to make the device visible to the instance (for example, `/dev/sdh` or `xvdh`). This attribute is reserved for internal use only.
NativeDevice	This attribute is reserved for internal use only.
NumThreads	The number of threads that are used within the agent process for managing resources. This number does not include the number of threads that are used for other internal purposes. Setting the NumThreads attribute to a higher value may decrease the time required to go online or the time required to monitor a large number of Amazon EBS volume resources. Type and dimension: static integer-scalar Default: 1

Resource type definition

```
type EBSVol (
        static int LevelTwoMonitorFreq = 3
        static int NumThreads = 1
        static str ArgList[] = { VolumeId, AWSBinDir, AWSDevice, NativeDevice }
        static boolean AEPTimeout = 1
        str VolumeId
```

```
        str AWSBinDir
        temp str AWSDevice
        temp str NativeDevice
)
```

Sample configuration

```
EBSVol ebsvol (
    VolumeId = vol-0c4bbfa4246964e73
    AWSBinDir = "/usr/local/bin/"
    )
```

Debug log levels

The EBSVol agent uses the following debug log levels:

DBG_1

AzureDisk agent

Virtual machines in Azure use data disks to store the applications data. The AzureDisk agent supports managed and unmanaged data disks and provides high availability of these disks during fail-over of an application.

The AzureDisk agent brings online, takes offline, and monitors the managed and unmanaged Azure data disks. It attaches the managed and unmanaged data disks to a virtual machine of the same or different resource group. The AzureDisk agent uses Azure Python SDK to determine whether the Azure data disks are attached to the Azure virtual machines or not.

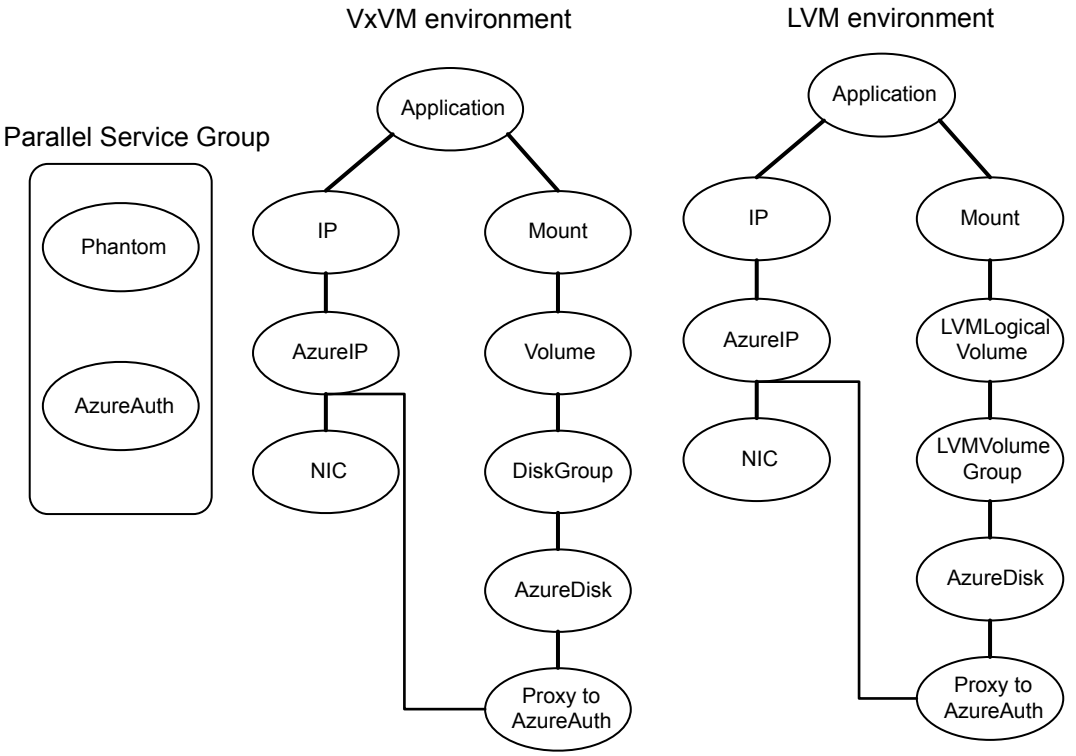
Prerequisites

- Configure AzureAuth agent. See “AzureAuth agent” on page 272.

Dependencies

The Azure disk resources depend on the AzureAuth resource.

Figure 2-13 Sample service group dependency



Agent functions

Open	Fetches the name of the Azure VM.
Online	Attaches Azure data disk to the Azure VM.
Offline	Detaches Azure data disk from the Azure VM.
Monitor	Determines if the Azure data disk is attached to the current Azure VM or not.
Clean	Terminates all ongoing resource actions and detaches the Azure data disk from the Azure VM.

State definitions

ONLINE	Indicates that the Azure data disk is attached to the Azure VM.
OFFLINE	Indicates that the Azure data disk is detached from the Azure VM.
FAULTED	Indicates that the Azure data disk was unexpectedly detached from the Azure VM outside of VCS control.
UNKNOWN	Indicates that a problem exists because of one of the following reasons: ■ Azure Python SDK is not installed ■ Azure VM is not found in the specified Resource Group ■ Azure data disk is not configured correctly or has invalid ID ■ Roles are not configured correctly or attached to the instance ■ The disks are attached to Azure VM of another cluster ■ DiskIds has a combination of managed and unmanaged disks ■ Configured unmanaged disks belong to different storage accounts

Attributes

Table 2-24 Required attributes

Attribute	Description
DiskIds	In case of managed disks, resource ID of the disk. In case of unmanaged disks, VHDUri of the disk. You can add one or more disk Ids separated by a space. Type and dimension: string-vector
VMResourceGroup	Azure resource group where the Azure VM resides. Type and dimension: string-scalar
AzureAuthResName	Name of the authentication agent resource that handles Azure related authentication. Type and dimension: string-scalar

Table 2-24 Required attributes *(continued)*

Attribute	Description
StorageAccountKey	Access key of the storage account. You must encrypt this key by using the vcsencrypt utility. For details, see the *Cluster Server Administrator's Guide*. This attribute is required only if unmanaged disks are configured. Type and dimension: string-scalar

Table 2-25 Optional attributes

Attribute	Description
AzureVMName	Name of the VM in Azure on which agent is running. Type and dimension: string-scalar

Resource type definition

```
type AzureDisk (
    static int MonitorTimeout = 120
    static str ArgList[] = { StorageAccountKey, tempVMName, DiskIds, VMResourceGroup,
        AzureVMName,"AzureAuthResName:SubscriptionId","AzureAuthResName:ClientId",
       "AzureAuthResName:SecretKey", "AzureAuthResName:TenantId" }
    str DiskIds[]
    str VMResourceGroup
    str AzureVMName
    str AzureAuthResName
    temp str tempVMName
    str StorageAccountKey
)
```

Sample configuration

Managed disks:

```
AzureDisk azure-res-disk (
DiskID = "/subscriptions/6940a326-fgh6-40dd-b616-c1e9bbdf1d63/resourceGroups/azureRG/providers/
          Microsoft.Compute/disks/AzureManagedDisk"
VMResourceGroup = "azureVMRG"
AzureAuthResName = "azure-auth-res"
```

```
)
```

Unmanaged disks:

```
AzureDisk azure-res-disk (
DiskIds = "https://azureunstorageaccount2.blob.core.windows.net/vhds/unmanageddisk.vhd"
VMResourceGroup = "azureVMRG"
AzureAuthResName = "azure-auth-res"
storageAccountKey = fpjNjrNrgRupNnnLhlKppPhnIpnNKpkRkrHnlRjpKnuLitIpjTgnTpfNglSls
)
```

Debug log levels

The AzureDisk agent uses the following debug log levels:

DBG_1

Chapter 3

Network agents

This chapter includes the following topics:

About the network agents

Use network agents to provide high availability for networking resources.

Note: The `ifconfig` command is deprecated, instead use the ip command for network operations. While the ifconfig and ip both are supported for backward compatibility, Veritas recommends that you use the ip command.

Agent comparisons

Agent comparisons may be made as described in the following sections.

The network agents support IPv4 and IPv6 addresses.

IP and NIC agents

The IP and NIC agents:

- Monitor a single NIC

IPMultiNIC and MultiNICA agents

The IPMultiNIC and MultiNICA agents:

- Operate in two modes:
  - IP Conservation Mode (ICM), which uses fewer IP addresses
  - Performance Mode (PM), which provides faster failover, but uses more IP addresses
- Monitor single or multiple NICs
- Check the backup NICs at fail over (ICM) or as soon as it comes up (PM)
- Use the original base IP address when failing over (ICM) or Require a pre-assigned base IP address for each NIC (PM)
- Have only one active NIC at a time in case of ICM and more than one active NIC at a time in case of PM

802.1Q trunking

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

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

```
# vconfig add eth2 10
```

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

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

VCS neither creates nor removes trunk interfaces. The administrator should set up the trunking as per the operating system vendor's documentation rather than using `vconfig` directly.

IP agent

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

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

For the NIC and IP agents, VCS supports Linux bonded interfaces.

High availability fire drill for IP agent

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

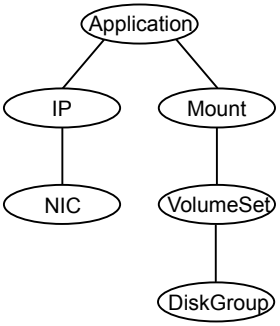
- Checks for the existence of a route to the IP from the specified NIC
- Checks for the existence of the interface configured in the IP resource

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

Dependencies for IP agent

IP resources depend on NIC resources.

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



Agent functions for IP agent

Online	Configures the IP address to the NIC. Checks if another system is using the configured IP address and issues a warning. For IPv4 addresses, it uses the ifconfig command to set the IPv4 address on an unique alias of the interface if the Options attribute is configured and none of the IPOptions and IPRouteOptions attributes are configured, else it makes use of the ip command. For IPv6 addresses, the ip command is used.
Action	■ route.vfd Checks for the existence of a route to the IP from the specified NIC. ■ device.vfd Checks for the existence of the interface configured in the Device attribute.
Offline	Brings down the IP address that is specified in the Address attribute.
Monitor	Monitors the interface to test if the IP address that is associated with the interface is alive.
Clean	Brings down the IP address that is specified in the Address attribute.

State definitions for IP agent

The state definitions for this agent follow:

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.
FAULTED	Indicates that the IP address could not be brought online, usually because the NIC configured in the IP resource is faulted or the IP address was removed out of VCS control.

Attributes for IP agent

Table 3-1 Required attributes

Required attribute	Description

Address	A virtual IP address that is associated with the interface, and which is different from the base IP address.
	Type and dimension: string-scalar
	Examples:
	■ IPv4: "192.203.47.61"
	■ IPv6: "2001::10"
	Note: For configuring the IP address of a different network than the network of the base IP address, you need to configure the following values in /etc/sysctl.conf.
	File: /etc/sysctl.conf
	`# avoid deleting secondary IPs on deleting the primary IP`
	`net.ipv4.conf.default.promote_secondaries = 1`
	`net.ipv4.conf.all.promote_secondaries = 1`
	After changing these values, load the current values of the /etc/sysctl.conf file using the following command: `# sysctl -p /etc/sysctl.conf`
	This is to make sure that if the primary IP address is unplumbed, the secondary IP address for a network is promoted to the primary address.

Device	The name of the NIC device that is associated with the IP address. Requires the device name without an alias. Type and dimension: string-scalar Example: eth0 In above example, eth0 is specified to assign the IP address to the next available alias of eth0. Use the `ip addr` command to display a list of NICs that are up and the IP addresses assigned to each NIC.

Table 3-1 Required attributes *(continued)*

Required attribute	Description
One of the following attributes: ■ NetMask: Mandatory only if you configure an IPv4 address. ■ PrefixLen: Mandatory only if you configure an IPv6 address.	
NetMask	The subnet mask that is associated with the IP address. For the IPv4 protocol, specify the value of NetMask attribute in decimal (base 10). Configure this attribute if the IP address is an IPv4 address. Type and dimension: string-scalar Example: "255.255.255.0"
PrefixLen	Prefix for the IPv6 address represented as the CIDR value. Type-dimension: integer-scalar Range: 0 - 128 Default: 1000 **Note:** Note that the default value is intentionally invalid for this attribute. You must set the value of this attribute to a range from 0 to 128 to activate this attribute. Example: 64

Table 3-2 Optional attributes

Optional attribute	Description
Options	Options for the `ifconfig` command. The ifconfig command generally resembles: `ifconfig` *`dev`* `inet` *`ipv4addr`* `netmask` *`netmask Options`* `up` You must configure either this Options attribute or both the IPOptions and IPRouteOptions attributes. Do not configure any options in this attribute that you can specify in other attributes. For example, for the netmask use the NetMask attribute. For complete list of ifconfig options refer to *ifconfig manpage*. Type and dimension: string-scalar Example: "broadcast 172.20.9.255"

Table 3-2 Optional attributes *(continued)*

Optional attribute	Description
IPOptions	Specifies the extra options that are passed to the `ip addr add` command. The agent uses this attribute in tandem with the IPRouteOptions attribute. The `ip addr add` command generally resembles: `"ip -4 addr add ipv4addr/prefixlen IPOptions label label dev device "` **Note:** If you configure this attribute, the agent ignores the Options attribute and uses the ip command instead of `ifconfig`. Type and dimension: string-scalar Examples: ■ "broadcast 172.20.9.255" ■ "scope link"
IPRouteOptions	Specifies the extra options that are passed to the `ip route add` command. The agent uses this attribute in tandem with the IPOptions attribute. The `ip route add` command resembles: `"ip route add IPRouteOptions dev device"` **Note:** If you configure this attribute, the agent ignores the Options attribute and uses the ip command instead of ifconfig. Type and dimension: string-scalar Examples: ■ "default via 172.20.9.1" ■ "scope link"

Resource type definition for IP agent

The resource definition for this agent on Linux follows:

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

```
        Options, IPOptions, IPRouteOptions }
        str Device
        str Address
        str NetMask
        int PrefixLen = 1000
        str Options
        str IPOptions
        str IPRouteOptions
)
```

Sample configurations for IP agent

The sample configurations for this agent follow:

IPv4 Configuration

Configuration 1 for Linux follows:

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

IPv6 Configuration

Configuration using a specified NetMask for Linux follows:

```
IP IP_2001_10 (
Device = eth0
Address = "2001::10"
PrefixLen = 64
)
```

Debug log levels for IP agent

The IP agent uses the following debug log levels:

DBG_1, DBG_2, DBG_4, DBG_5

NIC agent

The NIC agent monitors the configured NIC. If a network link fails, or if a problem arises with the NIC, the resource is marked FAULTED. You can use the agent to

make a single IP address on a single adapter highly available. This resource's Operation value is None.

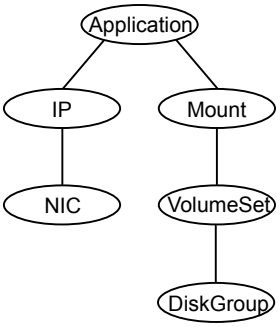
Some NICs maintain their connection status in a hardware register. For NICs that maintain their connection status, the agent uses ethtool and MII to determine the status of the NIC resource. For NICs that do not maintain their connection status, the agent uses a ping or a broadcast to determine the status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

For the NIC and IP agents, VCS supports Linux bonded interfaces.

Dependencies for NIC agent

The NIC resource does not depend on any other resources.

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



Bonded network interfaces for NIC agent

The NIC agent now supports using bonded network interfaces.

See "Monitoring bonded NICs for NIC agent" on page 110.

Agent functions for NIC agent

Monitor	■ If the NIC maintains its connection status, the agent uses MII to determine the status of the resource. If the NIC does not maintain its connection status, the agent verifies that the NIC is configured. The agent then sends a ping to all the hosts that are listed in the NetworkHosts attribute. If the ping test is successful, it marks the NIC resource ONLINE. If the NetworkHosts attribute list is empty, or the ping test fails, the agent counts the number of packets that the NIC received. The agent compares the count with a previously stored value. If the packet count increases, the resource is marked ONLINE. If the count remains unchanged, the agent sends a ping to the broadcast address of the device to generate traffic on the network. The agent counts the number of packets that the NIC receives before and after the broadcast. If the count increases, the resource is marked ONLINE. If the count remains the same or decreases over a period of five broadcast cycles, the resource faults.
Action	■ device.vfd Checks for the existence of the interface configured in the Device attribute.

Note: The NIC agent supports ethtool and MII-based device status monitoring. If the Mii attribute for a NIC resource is set to 1, the agent first checks for the device status using ethtool and if it fails, it checks for the MII status for the device. The agent does not do any ping and packet count-based monitoring if it can successfully decide the status of the NIC device based on ethtool and MII tests.

State definitions for NIC agent

The state definitions for this agent follow:

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 for NIC agent

Table 3-3 Required attributes

Required attribute	Description
Device	Specifies the name of the NIC that you want to monitor. Use the `ip addr` command to list all network adapters and the IP addresses assigned to each NIC. Type and dimension: string-scalar Example: "eth0" or "eth1"

Table 3-4 Optional attributes

Optional attribute	Description
Mii	Flag that defines whether the NIC maintains its connection status. If this flag is set to 1, the agent uses ethtool and MII hardware registers, instead of the ping and packet count method. The agent uses this method to determine the health of the network card. If the flag is set to 0, the agent does not use ethtool and Mii to monitor the status of the NIC. Type and dimension: integer-scalar Default: 1

Table 3-4 Optional attributes *(continued)*

Optional attribute	Description

NetworkHosts	List of hosts on the network that are pinged to determine the state of the NIC. Specify the IP address of the host—not the host name.
	The specified hosts must be pingable:
	■ from all the cluster nodes that are specified in the SystemList attribute for the service group to which the resource belongs
	■ through the device that is specified in the Device attribute
	The command to ping the host (hostip) via a NIC device (nicdev) is:
	For IPv4:
	`# ping -I device hostip`
	For IPv6:
	`# ping6 -I device hostip`
	If more than one network host is listed, the monitor returns ONLINE if the ping test is successful with at least one of the hosts.
	You can use both IPv4 and IPv6 NetworkHost addresses, and you can configure both types of addresses in the same resource.
	Type and dimension: string-vector
	Example:
	IPv4:
	`{ "166.93.2.1", "166.99.1.2" }`
	IPv6:
	`{ "2001::1" , "166.93.2.1" }`

PingOptimize	Attribute that defines whether the agent sends a broadcast ping before it retrieves the received packet statistics. This attribute is used when Mii is not set and no network hosts are specified. If the value of this attribute is 1, the agent retrieves received packet statistics from the netstat command and compare them with previously stored values. The agent sends a broadcast ping to the network only if the packet count remains unchanged. If the value of this attribute is 0, the agent sends a broadcast ping before it checks the network statistics. Type and dimension: integer-scalar Default: 1

Resource type definition for NIC agent

The resource definition for this agent on Linux follows:

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

Notes for the NIC agent

The NIC agent has the following notes:

- High availability fire drill for NIC agent
- Monitoring bonded NICs for NIC agent
- Setting Mii and miimon for NIC agent

Monitoring bonded NICs for NIC agent

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

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

- The value of the miimon parameter, which you set while loading the bonding driver. miimon is a parameter to the bonding module and has a default setting of 0.
- The value of the Mii attribute (Mii) of the NIC resource, which you set at runtime. Mii is an attribute of the NIC resource and has a default setting of 1.

Setting Mii and miimon for NIC agent

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

High availability fire drill for NIC agent

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

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

Case 1

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

Case 2

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

Case 3

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

Sample configurations for NIC agent

Configuration for using Mii for NIC agent

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

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

Configuration for using network hosts for NIC agent

```
NIC groupx_eth0 (
Device = eth0
```

```
NetworkHosts = { "166.93.2.1", "166.99.1.2" }
)
```

IPv6 configuration for NIC agent

The following is a basic configuration for IPv6 with IP and NIC resources.

```
group nic_group (
SystemList = { sysA = 0, sysB = 1 }
Parallel = 1
)
NIC nic_resource (
Device@sysA = eth0
Device@sysB = eth1
PingOptimize = 0
NetworkHosts@sysA = { "2001:db8:c18:2:214:4fff:fe96:11",
"2001:db8:c18:2:214:4fff:fe96:1" }
NetworkHosts@sysB = { "2001:db8:c18:2:214:4fff:fe96:1111",
"2001:db8:c18:2:214:4fff:fe96:111" }
)
Phantom phantom_resource (
)
group ip_group (
SystemList = { sysA = 0, sysB = 1 }
)
IP ip_resource (
Device@sysA = eth0
Device@sysB = eth1
Address = "2001:db8:c18:2:214:4fff:fe96:102"
PrefixLen = 64
)
Proxy proxy_resource (
TargetResName = nic_resource
)
ip_resource requires proxy_resource
```

Debug log levels for NIC agent

The NIC agent uses the following debug log levels:

DBG_1, DBG_4, DBG_5

IPMultiNIC agent

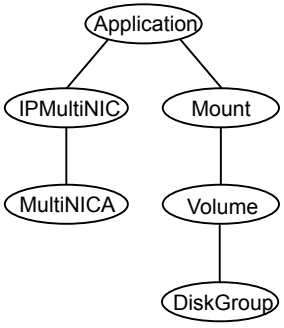
The IPMultiNIC agent manages the virtual IP address that is configured as an alias on one interface of a MultiNICA resource. If the interface faults, the agent works with the MultiNICA resource to fail over the virtual IP to a backup interface. If multiple service groups have IPMultiNIC resources associated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

For the IPMultiNIC and MultiNICA agents, VCS supports Linux bonded interfaces.

Dependencies for IPMultiNIC agent

IPMultiNIC resources depend on MultiNICA resources.

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



Agent functions for IPMultiNIC agent

Online	Configures a virtual IP address on the active interface of the MultiNICA resource. Also sends out a gratuitous ARP.
Offline	Removes the virtual IP address from the active interface of the MultiNICA resource.
Monitor	Checks if the virtual IP address is configured on one interface of the MultiNICA resource.
Clean	Removes the virtual IP address from active interface of the MultiNICA resource.

State definitions for IPMultiNIC agent

The state definitions for this agent follow:

ONLINE	Indicates that the specified IP address is assigned to one of the interfaces specified in the corresponding MultiNICA resource.
OFFLINE	Indicates that the specified IP address is not assigned to any interface of the MultiNICA resource.
UNKNOWN	Indicates that the agent can not determine the state of the resource. This state may be due to an incorrect configuration.
FAULTED	Indicates that the IP address could not be brought online, usually because all the NICs in the MultiNICA resource are faulted or the IP address was removed out of VCS control.

Attributes for IPMultiNIC agent

Table 3-5 Required attributes

Required attribute	Description
Address	The virtual IP address that is assigned to the active NIC. Type and dimension: string-scalar Examples: ■ IPv4: "10.128.10.14" ■ IPv6: "2001:DB8::"
MultiNICAResName	Name of the associated MultiNICA resource that determines the active NIC. Type and dimension: string-scalar Example: "mnic"
One of the following attributes: ■ NetMask: Mandatory only if you configure an IPv4 address. ■ PrefixLen: Mandatory only if you configure an IPv6 address.	
NetMask	For the IPv4 protocol, specify the value of NetMask attribute in decimal (base 10). Configure this attribute if the IP address is an IPv4 address. Type and dimension: string-scalar Example: "255.255.255.0"

Table 3-5 Required attributes *(continued)*

Required attribute	Description
PrefixLen	Specifies the prefix for the IPv6 address represented as the CIDR value. When you use the IPv6 protocol, you must configure a value for this attribute. Type-dimension: integer-scalar Range: 0 - 128 Default: 1000 Note: The default value is intentionally invalid for this attribute. You must set the value of this attribute to a range from 0 to 128 to activate this attribute. Example: 64

Table 3-6 Optional attributes

Optional attribute	Description
Options	The `ifconfig` command options for the virtual IP address. Do not configure any options in this attribute that you can specify in other attributes. For example, for the netmask use the NetMask attribute. Type and dimension: string-scalar Example: "mtu 2000"
IPOptions	Specifies the extra options that are passed to the `ip addr add` command. The `ip addr add` command resembles the following: ■ IPv4 "`ip addr add` *`ipv4addr/prefixlen IPOptions`* `label` *`label`* `dev` *`device`*" ■ IPv6 "`ip addr add` *`ipv6addr/prefixlen IPOptions`* `label` *`label`* `dev` *`device`*" Type and dimension: string-scalar **Note:** If you configure this attribute, the agent ignores the Options attribute and uses the ip command instead of ifconfig. Example: ■ "broadcast 172.20.9.255" ■ "scope link"

Resource type definition for IPMultiNIC agent

The resource definition for this agent on Linux follows:

```
type IPMultiNIC (
static int MonitorTimeout = 200
static int OfflineMonitorInterval = 120
static int ToleranceLimit = 2
static str ArgList[] = { Address, NetMask, PrefixLen,
MultiNICAResName, Options, IPOptions, "MultiNICAResName:Probed"
}
str Address
str MultiNICAResName
str NetMask
int PrefixLen = 1000
str Options
str IPOptions
)
```

Sample configuration: IPMultiNIC and MultiNICA

Refer to the MultiNICA agent for more information.

IPv4 configuration for IPMultiNIC agent

```
cluster foo (
UserNames = { admin = "cDRpdxPmHpzS." }
CounterInterval = 5
)
system sysA (
)
system sysB (
)
group grp1 (
SystemList = { sysA = 1, sysB = 2 }
)
IPMultiNIC ip1 (
Address = "192.123.10.177"
MultiNICAResName = mnic
NetMask = "255.255.248.0"
)
MultiNICA mnic (
Device @sysA = { eth0 = "192.123.10.127", eth1 =
```

```
"192.123.11.127" }
Device @sysB = { eth0 = "192.123.10.128", eth2 =
"192.123.11.128" }
NetMask = "255.255.248.0"
NetworkHosts = { "192.123.10.129", "192.123.10.130" }
)
ip1 requires mnic
// resource dependency tree
//
// group grp1
// {
// IPMultiNIC ip1
// {
// MultiNICA mnic
// }
// }
```

IPv6 configuration for IPMultiNIC agent

```
cluster foo (
UserNames = { admin = "cDRpdxPmHpzS." }
CounterInterval = 5
)
system sysA (
)
system sysB (
)
group grp1 (
SystemList = { sysA = 1, sysB = 2 }
)
IPMultiNIC ip1 (
Address = "2001::110"
MultiNICAResName = mnic
PrefixLen=96
)
MultiNICA mnic (
Device @sysA = { eth0 = "2001::10", eth1 = "2001::11" }
Device @sysB = { eth0 = "2001::12", eth2 = "2001::13" }
PrefixLen=96
NetworkHosts = { "2001::1", "2001::500" }
)
ip1 requires mnic
// resource dependency tree
```

```
//
// group grp1
// {
//     IPMultiNIC ip1
//     {
//     MultiNICA mnic
//      }
// }
```

Mixed mode configuration—IPv4 and IPv6 for IPMultiNIC agent

Mixed mode configuration for IPv4 and IPv6 follows:

```
cluster foo (
UserNames = { admin = "cDRpdxPmHpzS." }
CounterInterval = 5
)
system sysA (
)
system sysB (
)
group grp1 (
SystemList = { sysA = 1, sysB = 2 }
)
IPMultiNIC ip1 (
Address = "2001::110"
MultiNICAResName = mnic
PrefixLen=96
)
IPMultiNIC ip2 (
Address = "192.123.10.177"
MultiNICAResName = mnic
NetMask="255.255.248.0"
)
MultiNICA mnic (
Device @sysA = { eth0 = "192.123.10.127", eth1 =
"192.123.11.128" }
Device @sysB = { eth0 = "192.123.10.129", eth2 =
"192.123.11.130" }
NetMask = "255.255.248.0"
DualDevice @sysA = { eth0 = "2001::10", eth1 =
```

```
"2001::11" }
DualDevice @sysB = { eth0 = "2001::12", eth2 =
"2001::13" }
PrefixLen=96
NetworkHosts = { "2001::1", "192.123.10.129" }
)

ip1 requires mnic
ip2 requires mnic
// resource dependency tree
//
// group grp1
// {
// IPMultiNIC ip1
// {
// MultiNICA mnic
// }
// IPMultiNIC ip2
// {
// MultiNICA mnic


// }
// }
```

Debug log levels

The IPMultiNIC agent uses the following debug log levels:

DBG_1, DBG_2, DBG_4, DBG_5

MultiNICA agent

The MultiNICA represents a set of network interfaces, and provides failover capabilities between them. You can use the agent to make IP addresses on multiple-adapter systems highly available.

The IPMultiNIC agent depends upon the MultiNICA agent to select the most preferred NIC on the system. IPMultiNIC brings the virtual IP online or offline. However, if the MultiNICA resource changes its active device, the MultiNICA agent handles the shifting of IP addresses.

If a NIC on a system fails, the MultiNICA agent selects another active NIC. The agent then shifts the virtual IP address to the newly selected active NIC. Only in a

case where all the NICs that form a MultiNICA agent fail, does the virtual IP address shift to another system.

If you associate an interface with a MultiNICA resource, do not associate it with any other MultiNICA or NIC resource.

If the same set of interfaces must be a part of multiple service groups, configure:

- A MultiNICA resource in one of the service groups, and
- The Proxy resources that point to the MultiNICA resource in the other service groups.

The MultiNICA agent can operate in two modes:

- IP Conservation Mode (ICM) for MultiNICA agent
- Performance Mode (PM) for MultiNICA agent

With sufficient IP addresses, use PM.

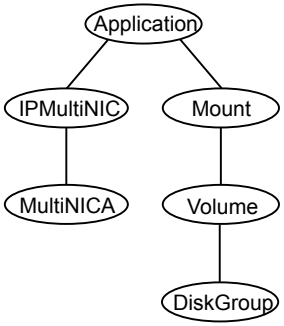
The IPMultiNIC and MultiNICA agents support Linux bonds.

The IPMultiNIC and MultiNICA agents support IPv4 and IPv6.

Dependencies for MultiNICA agent

The MultiNICA resource does not depend on any other resources.

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



IP Conservation Mode (ICM) for MultiNICA agent

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

Configuration for MultiNICA agent

When a MultiNICA resource is to be configured in ICM, the configured NICs must have the same base IP address. This IP address must be unique, and cannot appear

on any other NIC on any other node. You do not need to enable the base IP addresses beforehand. This mode does not support failing back the NIC, see the optional Failback attribute.

Operation for MultiNICA agent

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

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

Performance Mode (PM) for MultiNICA agent

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

Configuration for MultiNICA agent

When the MultiNICA resource is to be configured in PM, each NIC must have a unique base IP address. The base IP address cannot appear on any other NIC on the same node or any other node. The base IP address of all the devices in a single MultiNICA resource must belong to the same subnet in the configuration.

When you configure a single NIC under a MultiNICA resource, the MultiNICA agent uses PM. The base IP addresses have to be enabled on each NIC under MultiNICA control before starting VCS and handing over the management of the NICs to the agent.

Operation for MultiNICA agent

The agent uses this mode when all NICs under the MultiNICA agent have separate base IP addresses specified.

The mode requires that you enable the base IP addresses before starting VCS. When a NIC goes down, the agent migrates only virtual IP addresses.

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

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

Agent function for MultiNICA agent

The agent functions for this agent follow:

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

Attributes for MultiNICA agent

While configuring the MultiNICA resource, in addition to the required attributes, you must also configure at least one set of attributes from the following:

- For IPv4 configurations:
  - The Options attribute and the RouteOptions attribute, or
  - The IPv4RouteOptions attribute
- For IPv6 configurations:
  - The IPv6RouteOptions attribute

Table 3-7 Required attributes

Required attribute	Description
Device	List of devices and associated base IP addresses. This attribute must be specified separately for each system in the SystemList. You must specify the devices in the list in the order of priority. The first device that the agent determines as "up" becomes the active device, to which the agent assigns a corresponding IP address. For IP Conservation Mode (ICM): if all the NICs configured in the Device attribute are down, the MultiNICA agent faults the resource after a 2-3 minute interval. This delay occurs because the MultiNICA agent tests the failed NIC several times before it marks the resource offline. The engine log records messages that provide a detailed description of the failover events. Find the engine log in /var/VRTSvcs/log/engine_A.log. For each system you must localize the attribute with a separate base IP address. Type and dimension: string-association Examples: ■ IPv4 Device@sys1={ eth1 = "10.212.100.178", eth2 = "10.212.100.179" } Device@sys2 ={ eth2 = "10.212.100.180", eth3 = "10.212.100.181" } ■ IPv6 Device@sys1={ eth1 = "1234::5678", eth2 = "1234::5679" } Device@sys2={ eth3 = "1234::5680", eth4 = "1234::5681"} You can use IPv4 or IPv6 base addresses in the Device attribute, but make sure all of the base addresses use a common IP version.
One of the following attributes: ■ NetMask: Mandatory only if you configure an IPv4 address. ■ PrefixLen: Mandatory only if you configure an IPv6 address.	
NetMask	Specifies the netmask that is associated with the base IP address. The value must be specified in decimal (base 10). Configure this attribute if the IP address is an IPv4 address. Type and dimension: string-scalar Example: "255.255.252.0"

Table 3-7 Required attributes *(continued)*

Required attribute	Description
PrefixLen	Specifies the prefix for the IPv6 address represented as the CIDR value. When you use the IPv6 protocol, you must configure a value for this attribute. Type-dimension: integer-scalar Range: 0 - 128 Default: 1000 **Note:** The default value is intentionally invalid for this attribute. You must set the value of this attribute to a range from 0 to 128 to activate this attribute. Example: 64

Table 3-8 Optional attributes

Optional attribute	Description
DualDevice	The DualDevice attribute specifies the list of devices and associated IPv6 base addresses. Specify: ■ this attribute separately for each system in the SystemList. ■ the devices in the list in the order of priority. The first device that the agent determines as "up" becomes the active device, to which the agent assigns a corresponding IP address. NICs in Device and DualDevice attributes should be identical and in the same order. Use the DualDevice attribute only when configuring mixed IPv4/IPv6 stacks. In that case, use the Device attribute to configure the IPv4 stack and the DualDevice attribute to configure the IPv6 stack. Example: DualDevice@sys1={ eth1 = 2001::DB8, eth2 = 2001::DB9} DualDevice@sys2={ eth3 = 2001::DB10, eth4 = 2001::DB11}
Failback	This attribute determines if the active NIC should be changed to a preferred NIC, even though the current NIC is healthy. If operating in the ICM mode, change the value to 0. Type and dimension: boolean-scalar Default: 1

Table 3-8 Optional attributes *(continued)*

Optional attribute	Description
IPv4AddrOptions	The extra options that are passed to the `ip addr add` command for IPv4 addresses.. The agent uses this attribute in tandem with the IPv4RouteOptions attribute. The `ip addr add` command generally resembles: `"ip addr add ipv4addr/prefixlen IPv4AddrOptions dev device"` **Note:** If you configure this attribute, the agent ignores the Options attribute and uses the ip command instead of ifconfig. Type and dimension: string-scalar Example: ■ "broadcast 172.20.9.255" ■ "scope link"
IPv4RouteOptions	The extra options that are passed to the `ip route add` command for IPv4 addresses. The agent uses this attribute in tandem with the IPv4AddrOptions attribute. The `ip route add` command generally resembles: `"ip route add IPv4RouteOptions dev device"` **Note:** If Options attribute is configured, and none of IPv4AddrOptions and LinkOptions is configured, the agent ignores this attribute. Type and dimension: string-scalar Example: ■ "default via 172.20.9.1" ■ "scope link"
IPv6AddrOptions	The extra options that are passed to the `ip addr add` command for IPv6 addresses. The agent uses this attribute in tandem with the IPv6RouteOptions attribute. The `ip addr add` command resembles: `"ip addr add ipv6addr/ prefixlen IPv6AddrOptions dev device"` Type and dimension: string-scalar Example: "scope link"

Table 3-8 Optional attributes *(continued)*

Optional attribute	Description
IPv6RouteOptions	The extra options that are passed to the `ip route add` command for IPv6 addresses. The `ip route add` command generally resembles: `"ip route add IPv6RouteOptions device dev"` Type and dimension: string-scalar Examples: ■ "default via 2001::DB2" ■ "scope link"
LinkOptions	Specifies options for the `ip link` command, which can bring an interface up or down. The `ip link` command generally resembles: `"ip link dev up LinkOptions"` **Note:** If you configure this attribute, the agent ignores the Options attribute and uses the ip command instead of ifconfig. Type and dimension: string-scalar Example: ■ "mtu 1024" ■ "broadcast 172.20.9.255"
NetworkHosts	List of hosts on the network that receive pings to determine the state of the NICs. Specify the IP address of the host, not the host name. Include the hosts that all the NICs in the Device list can reach. If more than one network host is listed, monitor returns ONLINE if the ping test is successful with at least one of the hosts. You can use both IPv4 and IPv6 NetworkHost addresses, and you can configure both types of addresses in the same resource. Type and dimension: string-vector Example: NetworkHosts = { "2001::1", "192.123.10.129" }

Table 3-8 Optional attributes *(continued)*

Optional attribute	Description
Options	The `ifconfig` options that you want to use when you assign the base IP address to the active device. You must configure either this Options attribute or both the IPv4AddrOptions and IPv4RouteOptions attributes. This attribute does not support IPv6. **Note:** If you configure any of the IPv4AddrOptions or LinkOptions attribute, the agent ignores Options attribute and uses ip command instead of ifconfig. Type and dimension: string-scalar Example: "broadcast 10.212.100.255"
PingOptimize	Determines whether or not a broadcast ping is sent before checking network statistics, which are used to determine the state of the NIC (if MII is not supported and the ping to NetworkHosts does not confirm the NIC is up.) A value of 1 indicates a broadcast ping does not occur, a value of 0 indicates a broadcast ping occurs. Type and dimension: integer-scalar Default: 1
RouteOptions	Assignment of a base IP address to a device followed by a route add command. The command takes the options specified by this attribute. RouteOptions are applicable only when configuring the local host as the default gateway. No routes are added if this string is set to NULL. The RouteOptions attribute works in tandem with the Options attribute. You must configure the Options attribute when you use this attribute or the agent ignores this attribute. Type and dimension: string-scalar Example: "default gw 166.98.16.103""
Mii	Flag that defines whether the NIC maintains its connection status. If this flag is set to 1, the agent uses ethtool and MII hardware registers, instead of the ping and packet count method. The agent uses this method to determine the health of the network card. If the flag is set to 0, the agent does not use ethtool and Mii to monitor the status of the NIC. Type and dimension: boolean-scalar Default: 1

Resource type definition for MultiNICA agent

```
type MultiNICA (
        static int MonitorTimeout = 240
        static str ArgList[] = { Device, DualDevice, NetMask,
        PrefixLen, Options, RouteOptions, PingOptimize,
        MonitorOnly, NetworkHosts, Failback, LinkOptions,
        IPv4AddrOptions, IPv6AddrOptions, IPv4RouteOptions,
        IPv6RouteOptions, Mii }
        static str Operations = None
        str Device{}
        str DualDevice{}
        str NetMask
        int PrefixLen = 1000
        str Options
        str RouteOptions
        str LinkOptions
        str IPv4AddrOptions
        str IPv6AddrOptions
        str IPv4RouteOptions
        str IPv6RouteOptions
        int PingOptimize = 1
        str NetworkHosts[]
        boolean Failback = 1
        boolean Mii = 1
        )
```

Sample configurations for MultiNICA agent

The sample configurations for the following agent are:

MultiNICA and IPMultiNIC Performance Mode configuration

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

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

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

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

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

    system sysA (
    )
    system sysB (
    )

group grp1 (
    SystemList = { sysA = 1, sysB = 2 }
    )

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

MultiNICA mnic (
    Device @sysA = { eth0 = "192.123.10.127", eth1 =
    "192.123.11.128" }
    Device @sysB = { eth0 = "192.123.10.129", eth2 =
    "192.123.11.130" }
    NetMask = "255.255.248.0"
    NetworkHosts = { "192.123.10.1", "192.123.10.2" }
    )

ip1 requires mnic
    // resource dependency tree
    //
    // group grp1
    //          {
```

```
    // IPMultiNIC ip1
    //          {
    //          MultiNICA mnic
    //          }
    //     }
)
```

MultiNICA and IPMultiNIC IP Conservation Mode Configuration

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

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

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

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

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

system sysA (
    )

system sysB (
    )

group grp1 (
    SystemList = { sysA = 1, sysB = 2 }
    )
IPMultiNIC ip1 (
```

```
    Address = "192.123.10.177"
    MultiNICAResName = mnic
    NetMask = "255.255.248.0"
    )

MultiNICA mnic (
    Device @sysA = { eth0 = "192.123.10.127", eth1 =
    "192.123.10.127" }
    Device @sysB = { eth0 = "192.123.10.128", eth2 =
    "192.123.10.128" }
    NetMask = "255.255.248.0"
    NetworkHosts = { "192.123.10.1", "192.123.10.2" }
    Failback = 0

    )

ip1 requires mnic

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

IPv6 configuration for MultiNICA agent

The following sample is for IPv6 use.

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

    IPMultiNIC ipmnic_res (
        Address = "2007:192::1627:161"
        MultiNICAResName = mnica_res
        PrefixLen = 64
        )

    MultiNICA mnica_res (
```

```
        Device @sysA = { eth0 = "fe80::214:4fff:fe96:ae0a",
        eth1 = "fe80::214:4fff:fe96:ae0a" }
        Device @sysB = { eth0 = "fe80::214:4fff:fe96:ae0b",
        eth2 = "fe80::214:4fff:fe96:ae0b" }
        Failback = 0
        PrefixLen = 64
        )

    ipmnic_res requires mnica_res
```

Mixed mode configuration—IPv4 and IPv6 for MultiNICA agent

Mixed mode configuration for IPv4 and IPv6 follows:

```
cluster foo (
UserNames = { admin = "cDRpdxPmHpzS." }
CounterInterval = 5
)
system sysA (
)
system sysB (
)
group grp1 (
SystemList = { sysA = 1, sysB = 2 }
)
IPMultiNIC ip1 (
Address = "2001::110"
MultiNICAResName = mnic
Failback = 0
PrefixLen=96
)
IPMultiNIC ip2 (
Address = "192.123.10.177"
MultiNICAResName = mnic
NetMask="255.255.248.0"
)
MultiNICA mnic (
Device @sysA = { eth0 = "192.123.10.127", eth1 =
"192.123.11.127" }
Device @sysB = { eth0 = "192.123.10.128", eth1 =
"192.123.11.128" }
NetMask = "255.255.248.0"
```

```
DualDevice @sysA = { eth0 = "2001::10", eth1 = "2001::10" }
DualDevice @sysB = { eth0 = "2001::11", eth1 = "2001::11" }
Failback=0
PrefixLen=96
NetworkHosts = { "2001::1", "192.123.10.1" }
)
ip1 requires mnic
ip2 requires mnic
// resource dependency tree
//
// group grp1
// {
// IPMultiNIC ip1
// {
// MultiNICA mnic
// }
// IPMultiNIC ip2
// {
// MultiNICA mnic
// }
// }
```

Debug log levels for MultiNICA agent

The MultiNICA agent uses the following debug log levels: DBG_1, DBG_2, DBG_3, DBG_4, DBG_5

DNS agent

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

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

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

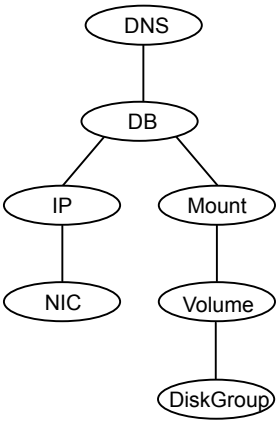
Use the DNS agent if the Resource Records need to be dynamically added and deleted from the DNS servers during failover. The agent updates the name server with the new resource record mappings while failing over and allows the clients to connect to the failed over instance of the application.

For important information about this agent, refer to Agent notes for DNS agent

Dependencies for DNS agent

No dependencies exist for the DNS resource.

Figure 3-5 Sample service group that includes a DNS resource



Agent functions for DNS agent

Online

Updates one or more name servers with the resource records.

The agent updates the name servers defined in the StealthMasters attribute. If you have not configured this attribute then the agent obtains the name of the master server by sending an Start of Authority (SOA) query. This query retrieves the SOA record of the zone defined in the agent's Domain attribute. This SOA record contains the name of the master server.

The agent creates PTR records for each RR of type A or AAAA if the value of the CreatePTR attribute is true. A prerequisite for this feature is that the same master or stealth server serves the forward (A or AAAA) and reverse zones.

Finally the agent generates an Online lock file to indicate that the resource is online on the current system.

Note: The DNS agent does not send any update for a resource record if it is already present on the name server.

Offline

Removes the Online lock file.

If attribute OffDelRR is true, offline removes all records that the ResRecord keys define.

Monitor	Returns the ONLINE state if at least one name server reports all mappings that ResRecord defines. The name servers are the master or StealthMaster servers and all the servers for which an NS record for the zone exists. The monitor entry point also sends periodic refresh requests to DNS server if the RefreshInterval attribute is set.
Clean	Removes the Online lock file, if it exists. If attribute OffDelRR is true, clean removes all records that the ResRecord keys define.
Open	Removes the Online lock file if the resource is reported online on another node inside the cluster to prevent concurrency violation. If the lock file exists, at least one name server has to report all the records that the ResRecord attribute defines. If all the name servers fail to report all the records, the agent function removes the Online lock file.
Action	Different action agent functions follow: ■ keyfile.vfd This action entry point checks if the key file as specified in the TSIGKeyFile attribute exists either locally or on shared storage. ■ dig.vfd This action entry point checks if dig and nsupdate binaries exist and are executable. ■ master.vfd This action entry point checks if stealth masters are able to reply to SOA query for the configured domain.

State definitions for DNS agent

The state definitions for this agent follow:

ONLINE	Online lock file exists and at least one name server can return all configured resource records.
OFFLINE	At least one of the following is true: ■ The online lock does not exist. ■ None of the name servers can report all of the RRs' mappings.
UNKNOWN	Indicates that the DNS resource is not configured correctly. Can indicate that the resource record list contains an invalid value as a part of the record key or a record value of the ResRecord attribute.

Attributes for DNS agent

Table 3-9 Required attributes

Required attribute	Description
Domain	A string representing the DNS zone that the agent administers. The domain name can only contain alphanumeric symbols and the dash. Type and dimension: string-scalar Examples: ■ Forward mapping: "demo.example.com" ■ IPv4 reverse mapping: "2.168.192.in-addr.arpa"

Table 3-9 Required attributes *(continued)*

Required attribute	Description
ResRecord	ResRecord is an association of DNS resource record values. Each ResRecord attribute consists of two values: *DNS record key* = *DNS record data*. Note that the record key must be a unique value. If the resource record list contains any invalid value as a part of the record key or a record data of the ResRecord attribute, the resource reports an UNKNOWN state. Type and dimension: string-association Examples: ■ For forward mapping, where the zone is demo.example.com: - sles901 = "192.168.2.191" - ww2 = sles901 - sles9ip6 = "2007::1:2:3:abc" ■ For a multi-home DNS record, typically for one host with two network interfaces and different addresses, but the same DNS name. The A type ResRecord configuration should be as follows: sle902 = "192.168.2.102 10.87.13.22" A multi-home AAAA DNS record can be configured as follows: sle902 = "1234::5678 1234::AABB:CCDD" ■ For reverse IPv4 address mapping, where the zone is 2.168.192.in-addr.arpa: 191 = "sles901.demo.example.com" ■ For reverse IPv6 address mapping, where the zone is 3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.7.0.0.2.ip6.arpa: cba = "sles9ip6.demo.example.com" Use only partial host names. If you use a fully qualified domain name, append a period "." at the end of the name. For CNAME records, use: ■ ResRecord = { www = mydesktop } or ■ ResRecord = { www = "mydesktop.marketing.example.com." } Where the Domain attribute is "marketing.example.com"

Table 3-9 Required attributes *(continued)*

Required attribute	Description
ResRecord (Continued)	The agent uses case-insensitive pattern matching—and a combination of the Domain and ResRecord attribute values—to determine the resource record type. The RR types are as follows: ■ PTR: if the Domain attribute ends with .arpa ■ A: if the record data field is an IPv4 address (four sets of numbers, where a period separates each set. The following details the pattern it tries to match: [*1-223*].[*0-255*].[*0-255*].[*0-255*] Hexadecimal is not supported.) ■ AAAA: if the record data fields are in multiple sets of hexadecimal format, then this record is an IPv6 associated type AAAA record. ■ CNAME: for any other valid record data. **Note:** If a name in the ResRecord attribute does not comply with RFC 1035, then the agent logs a warning message to the engine log file. This ResRecord association is not used. As an exception to this, the DNS agent allows underscore character ("_") in hostnames. Make sure that the DNS server supports the underscore character before you configure any DNS resource records to have the underscore character in their hostnames.

Table 3-10 Optional attributes

Optional attribute	Description
TTL	This attribute (a non-zero integer) represents the Time To Live (TTL) value, in seconds, for the DNS entries in the zone that you want to update. A lower value means more hits on your DNS server, while a higher value means more time for your clients to learn about changes. The TTL may take the value 0, which indicates never caching the record, to a maximum of 2,147,483,647, which is over 68 years! The current best practice recommendation (RFC 1912) proposes a value greater than one day, and on RRs that do not change often, consider multi-week values. Type and dimension: integer-scalar Default: 86400 Example: 3600
StealthMasters	The list of primary master name servers in the domain. This attribute is optional since the first name server is retrieved from the zone's SOA (Start of Authority) record. If the primary master name server is a stealth server, define this attribute. A stealth server is a name server that is authoritative for a zone, but does not appear in that zone's SOA record. It is hidden to prevent direct attacks from the Internet. Type and dimension: string-vector 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. This attribute should be configured only when the DNS server configured is a Unix based DNS server. Type and dimension: string-scalar Example: /var/tsig/example.com.+157+00000.private

Table 3-10 Optional attributes *(continued)*

Optional attribute	Description
CreatePTR	Use the CreatePTR attribute to direct the online agent functions to create PTR records for each RR of type A or AAAA. You must set the value of this attribute to true (1) to create the records. Before you can use this attribute, make sure that the same master or stealth servers serve the forward (A or AAAA) and reverse zones. Type and dimension: boolean-scalar Default: 0 Example: 1
OffDelRR	Use the OffDelRR attribute to direct the offline and clean agent functions to remove all records that the ResRecord key defines. You must set the value of this attribute to 1 (true) to have the agent remove all the records. Type and dimension: boolean-scalar Default: 0 Example: 1
UseGSSAPI	Use the UseGSSAPI attribute if the DNS server that you have configured is a Windows DNS server and only if it accepts secure dynamic updates. **Note:** Do not set this attribute if the Windows DNS server accepts non-secure updates. If this attribute is set to 1, the agent uses the -g option with the nsupdate command. See "Agent notes for DNS agent" on page 142. for more information on requirements to use the DNS agent with the secure Windows DNS server. Type and dimension: boolean-scalar Default: 0 Example: 1

Table 3-10 Optional attributes *(continued)*

Optional attribute	Description
RefreshInterval	This attribute represents the time interval in seconds after which the DNS agent attempts to refresh the resource records (RRs) on the DNS servers. The default value of zero indicates that the DNS agent does not attempt to refresh the records on the DNS servers. The DNS agent writes the warning message to the logs if it is not able to refresh the DNS records. **Note:** The refresh request is sent in the next monitor cycle after the RefreshInterval period is reached. If the DNS agent is unable to refresh the DNS records, and the records are removed as a result of a scavenging operation or by the DNS administrator, the DNS resource will fault. Type and dimension: integer-scalar Default: 0 Example: 3600
CleanRRKeys	Use this attribute to direct the online agent function to clean up all the existing DNS records for the configured keys before adding new records. The default value (0) disables this behavior. **Note:** If multiple DNS resources are configured with the same key value in their ResRecord attribute, then do not set this attribute value to 1. Type and dimension: boolean-scalar Default: 0 Example: 1

Resource type definition for DNS agent

The resource definition for this agent on Linux follows:

```
type DNS (
        static keylist SupportedActions = { "dig.vfd",
        "master.vfd", "keyfile.vfd" }
        static str ArgList[] = { Domain, TTL, TSIGKeyFile,
        StealthMasters, ResRecord, CreatePTR, OffDelRR,
        UseGSSAPI, RefreshInterval, CleanRRKeys }
        str Domain
        int TTL = 86400
```

```
        str TSIGKeyFile
        str StealthMasters[]
        str ResRecord{}
        boolean CreatePTR = 0
        boolean OffDelRR = 0
        boolean UseGSSAPI = 0
        int RefreshInterval = 0
        boolean CleanRRKeys = 0
)
```

Agent notes for DNS agent

The DNS agent has the following notes:

- About using the VCS DNS agent on UNIX with a secure Windows DNS server
- High availability fire drill for DNS agent
- Monitor scenarios for DNS agent
- Sample Web server configuration for DNS agent
- Secure DNS update for BIND 9 for DNS agent
- Setting up secure updates using TSIG keys for BIND 9 for DNS agent

About using the VCS DNS agent on UNIX with a secure Windows DNS server

This section describes the requirements for using the DNS agent with a secure Windows DNS server. Note that there are no special requirements for sending non-secure updates to a Windows DNS server.

Software requirement for DNS agent

For the secure updates on Windows DNS server to work, the VCS DNS agent on UNIX requires BIND version 9.7.2-P3 or later installed on all cluster nodes.

Configuration requirement for DNS agent

The VCS DNS agent on UNIX requires setting up Kerberos authentication with the Windows DNS server and configuring the domain and DNS server information in /etc/resolv.conf at the client node.

To set up the Kerberos authentication from the UNIX host to the Windows DNS server, configure the Kerberos configuration file (/etc/krb5.conf or /etc/ krb/krb5.conf) to use the Windows DNS server as Key Distribution Centre (KDC).

A sample Kerberos configuration file with domain privdns.sym and DNS server master.privdns.sym is as follows:

```
[libdefaults]
default_realm = PRIVDNS.SYM
dns_lookup_realm = true
dns_lookup_kdc = true
default_tkt_enctypes = des-cbc-md5
default_tgs_enctypes = des-cbc-md5
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true
allow_weak_crypto = true
[realms]
PRIVDNS.SYM = {
kdc = master.privdns.sym:88
kpasswd_server = master.privdns.sym:464
admin_server = master.privdns.sym
}
[domain_realm]
.privdns.sym = PRIVDNS.SYM
privdns.sym = PRIVDNS.SYM
```

Note: The DNS agent does not support KDC and Domain Controller/DNS located on different servers.

Authenticate all the nodes on the cluster (on which the DNS agent is configured to run) with the Active directory. Use kinit on your user account and use klist to verify that you have a ticket to the configured realm principal. Refer to the man page of kinit for more information on obtaining Kerberos ticket granting tickets from KDC.

Note: The DNS agent requires a node to be authenticated with Kerberos all the time. Renew the obtained tickets periodically if your authentication method requires you to do so.

A sample run of kinit and klist for the above configuration with user vcsdns will look as follows:

```
# kinit vcsdns
Password for vcsdns@PRIVDNS.SYM:
# klist
Ticket cache: FILE:/tmp/krb5cc_0
```

```
Default principal: vcsdns@PRIVDNS.SYM
Valid starting Expires Service principal
12/14/09 16:17:37 12/15/09 02:19:09 krbtgt/PRIVDNS.SYM@PRIVDNS.SYM
renew until 12/21/09 16:17:37
```

If the environment variable KRB5CCNAME is set to some non-default location (default is `/tmp`), then VCS will not inherit it by default and will look for the Kerberos tickets in default location `/tmp`.

To resolve this issue, un-set the environment variable KRB5CCNAME and run the kinit command again. This will update the Kerberos tickets in default location (`/tmp`). Else, for a customized location (for example, `/cache/krb_ticket`) for Kerberos tickets, add an entry in `/opt/VRTSvcs/bin/vcsenv` file on each cluster node before VCS starts:

```
KRB5CCNAME="FILE:/cache/krb_ticket"

export KRB5CCNAME
```

Update `/etc/resolv.conf` on your client node to add information for the Windows DNS server and the configured domain.

High availability fire drill for DNS agent

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

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

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

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

Monitor scenarios for DNS agent

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

Table 3-11 summarizes the monitor scenarios for the Online lock files.

Table 3-11 Monitor scenarios for the Online lock file

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

Sample Web server configuration for DNS agent

Take the example of a Web server. A browser requests the URL http://www.example.com that maps to the canonical name server1.example.com. The browser retrieves the IP address for the web server by querying a domain name server. If the web server fails over from server one to server two (server2.example.com), the domain name servers need a new canonical name mapping for www.example.com. After the failover, the DNS resource updates this mapping of www.example.com to point to canonical name server2.example.com

Note: In this configuration, the Domain attribute should be configured with value "example.com"

Secure DNS update for BIND 9 for DNS agent

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

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

Setting up secure updates using TSIG keys for BIND 9 for DNS agent

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

To use secure updates using TSIG keys, perform the following steps at the DNS server:

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

   ```
   # dnssec-keygen -a HMAC-MD5 -b 128 -n HOST example.com.
   ```

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

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

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

   ```
   +Cdjlkef9ZTSeixERZ433Q==
   ```

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

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

   Where `+Cdjlkef9ZTSeixERZ433Q==` is the key.

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

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

6. Save and restart the named process.

7 Place the files containing the keys on each of the nodes that are listed in your group's SystemList. The DNS agent uses this key to update the name server.

Copy both the private and public key files on to the node. A good location is in the /var/tsig/ directory.

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

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

Sample configurations for DNS agent

This section contains sample configurations for this agent.

Basic IPv6 configuration for DNS agent

This sample configuration provides basic configuration for IPv6 support. In the following configuration, *nic_value* represents the base NIC value for the platform

For example: eth0

```
group ipv6_group_dns (
    SystemList = { sysA = 0, sysB = 1 }
    )

    DNS ipv6group_dns_res (
        Critical = 0
        Domain = "example.com"
        TSIGKeyFile = "/var/tsig/Kipv6.vcscd.net.+157+18435.private"
        StealthMasters = { "2001:db8:c18:2:69c4:3251:bac1:6cbe" }
        ResRecord = {
            vcssysCv6 = "2001:db8:c18:2:214:4fff:fe96:8833",
            sysC = vcssysCv6 }
        )

    IP ipv6group_ip_res (
        Device @sysA = nic_value
        Device @sysB = nic_value
        Address = "2001:db8:c18:2:214:4fff:fe96:8833"
        PrefixLen = 64
```

```
        )

    NIC ipv6group_nic_res (
        Device @sysA = nic_value
        Device @sysB = nic_value
        NetworkHosts = { "2001:db8:c18:2:214:4fff:fea2:fd50" }


)

    ipv6group_dns_res requires ipv6group_ip_res
    ipv6group_ip_res requires ipv6group_nic_res
```

IPv6 CNAME sample configuration for DNS agent

The following sample configuration uses CNAME values.

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

    DNS cname_group_dns_res (
        Domain = "example.com"
        StealthMasters = { "3ffe:556::1000:5761" }
        ResRecord @sysA = { www = server1 }
        ResRecord @sysB = { www = server2 }
        OffDelRR = 1
    )
```

IPv4 A sample configuration for DNS agent

The following sample configuration uses A values.

```
group forwardv4_group (
    SystemList = { sysA = 0, sysB = 1 }
    )

    DNS forward_group_v4_resource (
        Domain = "example.com"
        StealthMasters = { "3ffe:556::1000:5761" }
        ResRecord @sysA = { www = "10.200.56.240" }
        ResRecord @sysB = { www = "10.200.56.244" }
        OffDelRR = 1
```

)

Debug log levels for DNS agent

The DNS agent uses the following debug log levels:

DBG_1, DBG_2, DBG_3, DBG_4, DBG_5

AWSIP agent

The AWSIP agent manages the networking resources in an Amazon Web Services (AWS) cloud environment. The following are the networking resources:

- Private IP—Private IP is a private numerical address that networked devices use to communicate with one another.
- Elastic IP—An Elastic IP address is a static IPv4 public address designed for dynamic cloud computing. An Elastic IP address is associated with your AWS account.
- Overlay IP—AWS allows you to redirect IP address traffic to an Elastic Compute Cloud (EC2) instance in a Virtual Private Network (VPC) no matter which subnet or availability zone (AZ) it is in. Overlay IP provides IP failover functionality for nodes spread across subnets or availability zones. Overlay IP must be outside of the VPC Classless Inter-Domain Routing (CIDR) block.

The agent uses AWS CLIs to associate IP resources in an AWS cloud environment. The agent does the following:

- Assigns and unassigns private IP address
- Associates and disassociates Elastic IP address and assigns/unassigns private IP
- Manages route table entries of overlay IP for failing over across subnets

The agent automatically fetches the Amazon EC2 region.

Prerequisites

- Install the AWS CLI package.
- Go to **EC2 instance** > **Networking** > **Change Source/Dest. Check** and disable **Change Source/Dest. Check** for overlay IP.
- Create an IAM role with requisite permissions and attach the roles to the EC2 cluster instances.

Go to **IAM** > **Create Policy** > **Role** and create the role.
The following is a sample of the policy .json file:

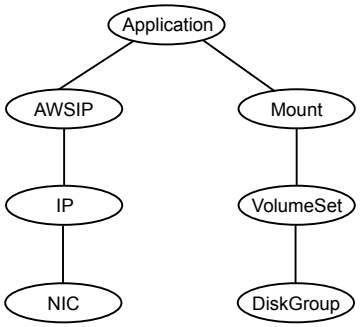
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:AssignPrivateIpAddresses",
                "ec2:DescribeAddresses",
                "ec2:DisassociateAddress",
                "ec2:AssociateAddress",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:AssignPrivateIpAddresses",
                "ec2:AssignIpv6Addresses",
                "ec2:UnassignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:CreateRoute",
                "ec2:DeleteRoute",
                "ec2:ReplaceRoute"
              ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

- Ensure that all outgoing traffic goes through network address translation (NAT) gateways. The agent uses AWS CLIs and requires access to the Amazon EC2 API endpoints.
- Ensure that the subnets in which the EC2 exists are associated with the route table.

Dependencies

IP resources depend on NIC and AWSIP resources.

Figure 3-6 Sample service group dependency



Agent functions

Online	■ Private IP: Assigns the private IP to a NIC Elastic Network Interface (ENI). ■ Elastic IP: Works only if a private IP is defined in the configuration. Associates the private IP with an elastic IP. Agent automatically detects the ENI based on the Device attribute. ■ Overlay IP: Creates a route in the route table given in the RouteTableIds attribute with destination as the OverlayIP and target as the ENI on which the entry point is running.
Offline	■ Private IP: Unassigns from the ENI. ■ Elastic IP: Disassociates the elastic IP. ■ Overlay IP: Deletes the route from the route table.
Monitor	■ Private IP: Monitors the assignment of the private IP to the ENI. ■ Elastic IP: Monitors the association between private IP and elastic IP. ■ Overlay IP: Monitors the routing information and checks whether the route table entry exists for the overlay IP.
Clean	■ Private IP: Unassigns the private IP. ■ Elastic IP: Disassociates the elastic IP and unassigns the private IP address. ■ Overlay IP: Deletes the route from the route table.

State definitions

ONLINE	■ Private IP: Private IP is assigned to the ENI. ■ Elastic IP: Private IP is assigned and elastic IP is associated with the private IP. ■ Overlay IP: Route table entry exists for the overlay IP.
OFFLINE	■ Private IP: Private IP is not assigned to the ENI. ■ Elastic IP: Private IP is not assigned or elastic IP is not associated with the private IP. ■ Overlay IP: Route table entry does not exist for the overlay IP.
UNKNOWN	If private IP, elastic IP, or overlay IP is in UNKNOWN state, one of the following could be true: ■ AWS CLI is not installed. ■ AWSPath attribute is not configured correctly. For example, if AWS CLI is installed in `/usr/local/bin/aws`, the AWSPath attribute should be `/usr/local/bin`. ■ Roles are not configured correctly or attached to the instance.
FAULTED	Indicates that the IP resources could not be brought online or abruptly stopped outside of VCS control.

Attributes

Table 3-12 Required attributes

Attribute	Description
PrivateIP	Secondary private IP address of the EC2 instance. Type and dimension: string-scalar
OverlayIP	Overlay IP provides IP failover functionality for nodes spread across subnets or availability zones. Overlay IP must be outside of the VPC CIDR block in which the nodes are present. Type and dimension: string-scalar

Table 3-13 Optional attributes

Attribute	Description
ElasticIP	An Elastic IP address is a static IPv4 address designed for dynamic cloud computing. An Elastic IP address is associated with your AWS account. You can map the Elastic IP address to a secondary private IP address. **Note:** This attribute does not support IPv6 address. Type and dimension: string-scalar
Device	Name of the network device. Enter `ifconfig -a` to list all network adapters. Type and dimension: string-scalar Example: eth0 In above example, eth0 is specified to assign the private IP address to the next available alias of eth0. One of the following attributes is mandatory: ■ PrivateIP ■ OverlayIP
RouteTableIds	Describes all the route tables. It can be one or more route table IDs. Type and dimension: string-list
AWSBinDir	Location of AWS EC2 commands and binaries. Type and dimension: string-scalar

Resource type definition

```
type AWSIP (
        static str ArgList[] = { PrivateIP, OverlayIP,
         ElasticIP, Device, RouteTableIds, AWSBinDir }
        str PrivateIP
        str OverlayIP
        str ElasticIP
        str Device
        keylist RouteTableIds
        str AWSBinDir
)
```

Samples configurations

Private IP

```
AWSIP jsrIP (
        PrivateIP = "2003::1"
        Device = eth0
        AWSBinDir = "/usr/local/bin"
        )
```

Elastic IP

```
AWSIP jsrIP (
        PrivateIP = "10.0.0.143"
        ElasticIP = "34.195.175.223"
        Device = eth0
        AWSBinDir = "/usr/local/bin"
          )
```

Overlay IP

```
AWSIP jsrIP (
        OverlayIP = "2010::1/128"
        Device = eth0
        AWSBinDir = "/usr/local/bin"
        RouteTableIds = { rtb-fb97ac9d, rtb-f416eb8d,
         rtb-e48be49d }
        )
```

AWSRoute53 agent

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. The AWSRoute53 agent updates and monitors the host name to IP address mapping. The agent does the mapping for the AWS Route 53 domain when failing over nodes across subnets. When you create a hosted zone, Amazon Route 53 automatically creates a name server (NS) record and a start of authority (SOA) record for the zone.

Use the AWSRoute53 agent if the Resource Records need to be dynamically added and deleted from the Route 53 domain during failover. The agent updates the new resource record mappings while failing over and allows the clients to connect to the failed over instance of the application.

Prerequisites

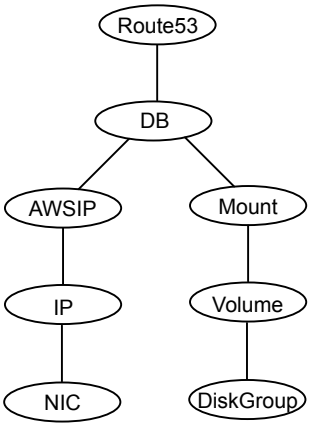
- Install the AWS CLI package.
- Create an IAM role with requisite permissions and attach the roles to the EC2 cluster instances.
  Go to **IAM** > **Create Policy** > **Role** and create the role.
  The following is a sample of the policy .json file:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "route53:ListResourceRecordSets",
                "route53:ChangeResourceRecordSets",
                "route53:GetHostedZone"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

- Ensure that all outgoing traffic goes through network address translation (NAT) gateways. The agent uses AWS CLIs and requires access to the Amazon EC2 API endpoints.

Dependencies

Figure 3-7 Sample service group dependency



Agent functions

Online	Updates Route 53 with the resource records. The agent updates the name servers defined in the Amazon Route 53 hosted zone.
Offline	Removes the Online lock file and removes the Route 53 record set from the Amazon Route 53 hosted zone if OffDelRR is set to 1. If OffDelRR is not set to 1, records are not deleted, but lock file is removed. Make sure you set OfflineTimeout and CleanTimeout to match the timeout accordingly. For example, if you have around 90 resource record sets in a resource, you should set the timeout to approximately 600.
Monitor	Returns the ONLINE state if all mappings are present in the AWS Route 53 hosted zone.
Clean	Removes the Online lock file and removes the Route 53 record set from the Amazon Route 53 hosted zone if OffDelRR is set to 1. If OffDelRR is not set to 1, records are not deleted, but lock file is removed. Make sure you set OfflineTimeout and CleanTimeout to match the timeout accordingly. For example, if you have around 90 resource record sets in a resource, you should set the timeout to approximately 600.

State definitions

ONLINE	Online lock file exists and all the resource record sets are present in the Route 53 hosted zone.
OFFLINE	Online lock does not exist and all or one of the resource record set(s) is not present in the Route 53 hosted zone.
UNKNOWN	The state will be UNKNOWN if one or more of the following is true: ■ AWS CLI is not installed. ■ AWSPath attribute is not configured properly. For example, if AWS CLI is installed in `/usr/local/bin/aws` directory, the AWSPath attribute should be `/usr/local/bin`. ■ Roles are not configured properly or attached to the instance. ■ HostedZoneID is not matching with the AWS Route 53 hosted zone ID. ■ Resource Record Type (RSType) is not A, CNAME, or PTR.

Attributes

Table 3-14 Required attributes

Attribute	Description
HostedZoneID	ID of the hosted zone containing the resource records. Type and dimension: string-scalar
ResRecord	ResRecord is an association of DNS resource record values. Each ResRecord attribute consists of two values: DNS record key = DNS record data. Note that the record key must be a unique value. If the resource record list contains any invalid value as a part of the record key or a record data of the ResRecord attribute, the resource reports an UNKNOWN state. Resource records can be of the following types: ■ A ■ CNAME ■ PTR Type and dimension: string-association

Table 3-15 Optional attributes

Attribute	Description
TTL	Specifies the minimum time to live (in seconds) for all resource records. Type and dimension: int-scalar
RSType	Amazon Route 53 supported DNS record types. Type and dimension : string-scalar
OffDelRR	Selects if resource records will be deleted as part of the resource offline entry point. Default is false. Type and dimension: boolean-scalar
AWSBinDir	Location of Amazon Route 53 commands and binaries. Type and dimension: string-scalar

Resource type definition

```
type AWSRoute53 (
        static int CleanTimeout = 600
        static int OfflineTimeout = 600
        static str ArgList[] = { HostedZoneID, TTL, RSType,
         OffDelRR, ResRecord, AWSBinDir }
        str HostedZoneID
        int TTL = 300
        str RSType = A
        boolean OffDelRR = 0
        str ResRecord{}
        str AWSBinDir
)
```

Sample configuration

A

```
AWSRoute53 awsroute53_A (
           HostedZoneID = Z1T9X503UWPVCM
           RSType = A
           ResRecord = { "xyz.com" = "40.1.1.1", "xyz1.com" = "30.1.1.1"}
     AWSBinDir = "/usr/local/bin"
                )
```

CNAME

```
AWSRoute53 awsroute53_cname (
           HostedZoneID = Z1WRLQVKRVUZPW
           RSType = CNAME
           OffDelRR = 1
           ResRecord = { "xyz.example.com" = "abc.example.com"}
      AWSBinDir = "/usr/local/bin/"
                )
```

PTR

```
AWSRoute53 awsroute53_ptr (
           HostedZoneID = Z1T9X503UWPVCM
           RSType = PTR
           OffDelRR = 1
           ResRecord = { 1 = "xyz.example.com"}
     AWSBinDir = "/usr/local/bin"
                )
```

AzureIP agent

The AzureIP agent manages the networking resources in an Azure environment.

The following are the networking resources:

- Private IP—Private IP is a private numerical address that networked devices use to communicate with one another. Private IP is used for communication within an Azure virtual network (VNet), and an on-premise network when you use a VPN gateway or ExpressRoute circuit to extend your network to Azure.
- Public IP— Public IP address is a numerical address that is used for communication with the Internet, including Azure public-facing services.
- Overlay IP—Overlay IP provides IP failover functionality for nodes spread across subnets. Overlay IP allows you to redirect IP address traffic to another cluster node belonging to different subnet within the same VNet. Overlay IP must be outside the VNet Classless Inter-Domain Routing (CIDR) block.

Note: OverlayIP does not work across Regions if VNet-to-VNet tunneling is used.

The agent uses Azure Python APIs to associate IP resources in a Azure VM.

The agent does the following:

- Gets the NIC details, create the IP configuration and associate and disassociates private IP address
- Associates and disassociates Public IP address with Private IP address
- Manages route table entries of overlay IP for failing over across subnets

An InfoScale deployment in Azure does not support IPv6 because of the following limitations:

- Existing virtual machines (VMs) that are configured with IPv4 addresses cannot use IPv6 addresses; you must deploy new VMs and configure them with IPv6 addresses.
- Public IPv6 addresses cannot be assigned to a VM.
- VMs with IPv6 addresses cannot be members of an Azure cloud service. However, they can communicate with each other over their respective IPv4 addresses.

Prerequisites

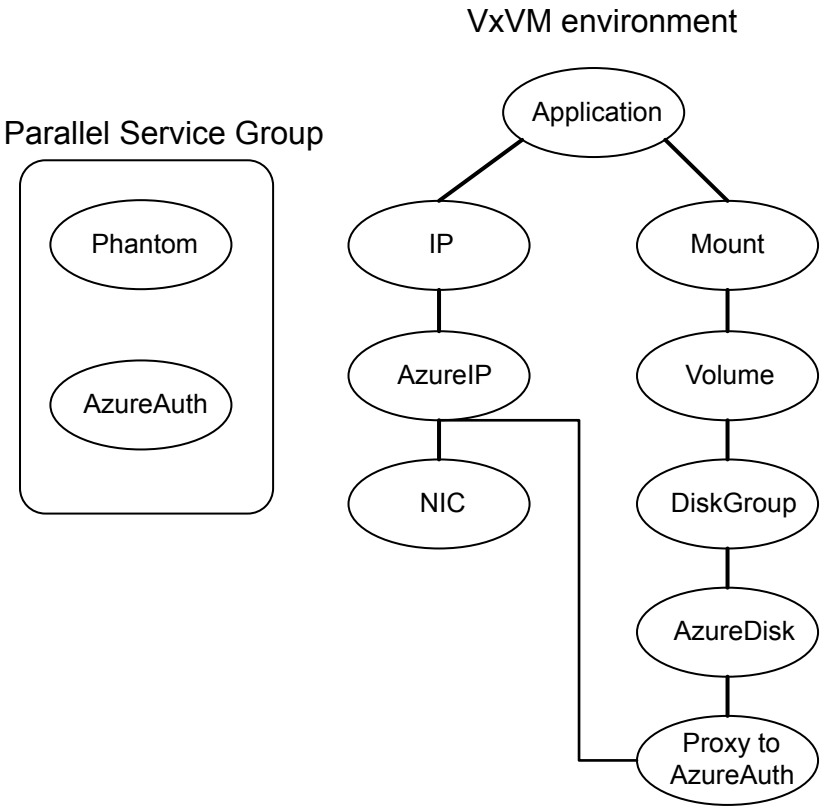
- Configure AzureAuth agent. See “AzureAuth agent” on page 272.
- To configure PublicIP, create static public IP resource in Azure portal.
- For Overlay IP, create a route table and associate the subnets, where the overlay IP fails over, with the route table.

Note: The subnets can be associated with a single route table.

Dependencies

The AzureIP resources depend on the AzureAuth resources.

Figure 3-8 Sample service group dependency



Agent functions

Online

- Private IP: Creates the IP configuration with the private IP and associates it with the Network Interface.
- Public IP: Creates the IP configuration with the private and public IP. Associates the IP configuration with the Network Interface.
- Overlay IP: Creates a route in the Azure route table given in the RouteTableResourceIds attribute with Address Prefix as the OverlayIP and next hop as the private IP of the primary NIC.

Offline/Clean	■ Private IP: Deletes the IP configuration to disassociate Private IP from the Network Interface. ■ Public IP: Deletes the IP configuration to disassociate the Public IP along with the Private IP from the Network Interface. ■ Overlay IP: Deletes the route from the route table.
Monitor	■ Private IP: Monitors the associations between IP configuration and Network Interface. ■ Public IP: Monitors the association between IP configuration and Network Interface. ■ Overlay IP: Monitors the routing information and checks whether the route table entry exists for the overlay IP.

State definitions

ONLINE	■ Private IP: Indicates that the Private IP is assigned to the NIC. ■ Public IP: Indicates that the Private IP is assigned and Public IP is associated with the private IP. ■ Overlay IP: Indicates that Route table entry exists for the overlay IP.
OFFLINE	■ Private IP: Indicates that the Private IP is not assigned to the NIC. ■ Public IP: Indicates that the Private IP is not assigned or Public IP is not associated with the private IP. ■ Overlay IP: Indicates that Route table entry does not exist for the overlay IP.
UNKNOWN	One of the following could be true: ■ Azure Python SDK is not installed. ■ Required privileges are not present to perform operations on Azure NIC and Virtual Machine. ■ Private IP, Public IP, or Overlay IP is specified in wrong format. ■ Private IP, Public IP, or Overlay IP is already in use. ■ NICDevice attribute value is invalid. ■ In case of Overlay IP, route table id(s) are invalid. ■ IP Configuration has invalid details on NIC. ■ Both or none of the Overlay IP and Private IP are set. Only one should be set.

FAULTED	Indicates that the IP resources could not be brought online or abruptly stopped outside of VCS control.

Attributes

Table 3-16 Required attributes

Attribute	Description
PrivateIP	Secondary private IP address of the Azure VM. This value is mandatory if OverlayIP is not provided. Type and dimension: string-scalar
NICDevice	Name of the network device. Enter `ip addr` to list all network adapters. Example: eth0 In above example, eth0 is specified to assign the private IP address to the next available alias of eth0. Type and dimension: string-scalar
OverlayIP	Overlay IP provides IP failover functionality for nodes spread across subnets. Overlay IP must be outside of the VNet CIDR block in which the nodes are present. This value is mandatory if PrivateIP is not provided. Type and dimension: string-scalar
RouteTableResourceIds	Describes all route tables. You can add one or more route table IDs separated by a space. Type and dimension: string-vector
AzureAuthResName	Name of the authentication agent resource that handles Azure related authentication. Type and dimension: string-scalar

Table 3-17 Optional attributes

Attribute	Description
PublicIP	Static public IP created from the Azure Portal. This IP is used in IP configuration to map the Public IP address to a secondary private IP address. Type and dimension: string-scalar
AzureVMName	Name of the VM in Azure on which agent is running. Type and dimension: string-scalar
VMResourceGroup	Azure resource group where the Azure VM resides. Type and dimension: string-scalar

Resource type definition

```
type AzureIP (
static str ArgList[] = { tempVMName, PrivateIP, NICDevice, PublicIP,
                tempPublicIPResourceId, AzureVMName, VMResourceGroup,
                OverlayIP, RouteTableResourceIds,
               "AzureAuthResName:SubscriptionId",
               "AzureAuthResName:ClientId", "AzureAuthResName:SecretKey",
               "AzureAuthResName:TenantId", tempVMResourceGroupName }
                str PrivateIP
                str NICDevice
                str PublicIP
                temp str tempPublicIPResourceId
                str AzureVMName
                str VMResourceGroup
                str OverlayIP
                str RouteTableResourceIds[]
                str AzureAuthResName
                temp str tempVMName
                temp str tempVMResourceGroupName
)
```

Sample configuration

Private IP

```
AzureIP azure-ip-res (
```

```
PrivateIP = "10.1.5.42"
NICDevice @ CLOUDVM1 = "eth0"
NICDevice @ CLOUDVM2 = "eth0"
AzureAuthResName = Auth_Res
)
```

Public IP

```
AzureIP azure-ip-res (
PrivateIP = "10.1.5.52"
NICDevice @ CLOUDVM1 = "eth0"
NICDevice @ CLOUDVM2 = "eth0"
PublicIP = "52.173.243.126"
AzureAuthResName = Auth_Res
)
```

OverLay IP

```
AzureIP overlay-ip-res (
NICDevice @ CLOUDVM1 = "eth0"
NICDevice @ CLOUDVM2 = "eth0"
OverlayIP = "192.168.3.88"
RouteTableResourceIds = {
         "/subscriptions/6940a326-abc6-40dd-b616-d3f9bbdf1d63/resourceGroups/azureRG/
         providers/Microsoft.Network/routeTables/azureroute1","/subscriptions
         /6940a326-abc6-40dd-b616-d3f9bbdf1d63/resourceGroups/azureRG/providers/
         Microsoft.Network/routeTables/azureroute2"}
AzureAuthResName = Auth_Res
)
```

Debug log levels

The AzureIP agent uses the following debug log levels:

DBG_1, DBG_2

AzureDNSZone agent

Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. Azure DNS lets you host your DNS domains on the cloud using Microsoft Azure infrastructure. The AzureDNSZone agent monitors and updates the host name to resource record mapping. The agent does the mapping for the Azure DNS domain when failing over nodes across subnets or regions. To

start hosting your domain in Azure DNS, you need to create a DNS zone for that domain name. When you create a DNS zone, Azure DNS automatically creates a name server (NS) record and a start of authority (SOA) record for the zone.

AzureDNSZone agent provides DNS-based traffic routing and failover. Use the AzureDNSZone agent if the resource records need to be dynamically added and deleted from the domain during failover. The agent updates the new resource record mappings while failing over and allows the clients to connect to the failed over instance of the application.

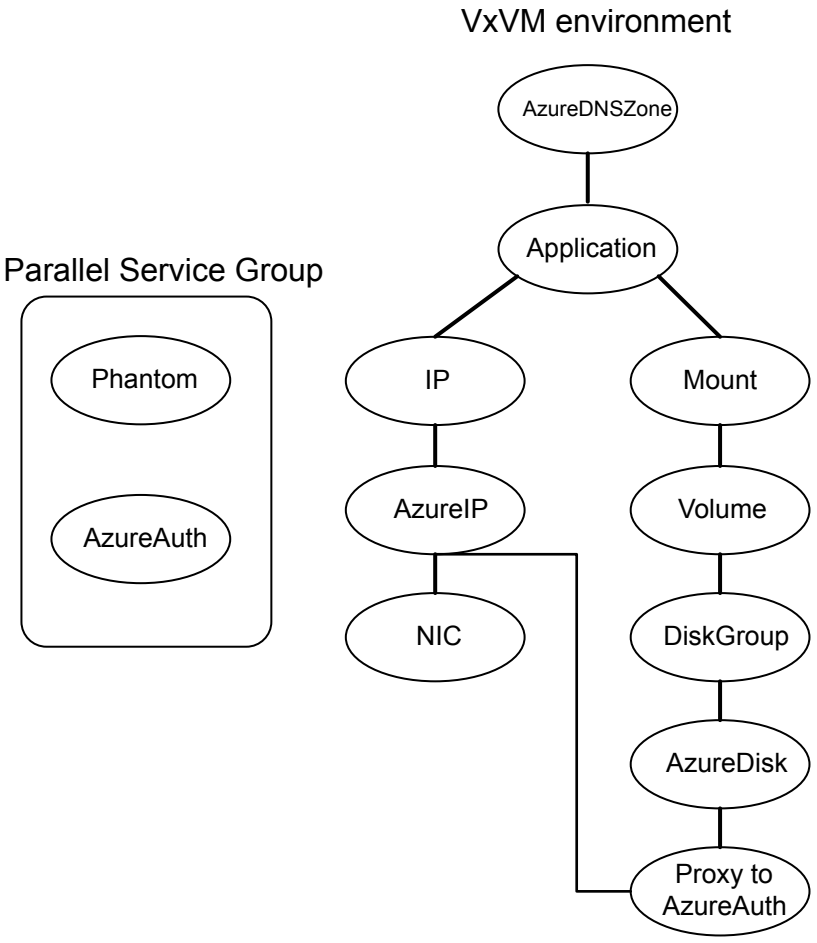
Prerequisites

- Configure AzureAuth agent. See “AzureAuth agent” on page 272.
- Own a domain name
- Create DNS zone with root domain name
- Delegate the domain name to Azure DNS. See “Delegating a domain to Azure DNS” on page 171.

Dependencies

The AzureDNSZone resources depend on the AzureAuth resource.

Figure 3-9 Sample service group dependency



Agent functions

Online	Creates or updates resource record sets in Azure DNS Zone.
Offline	Removes the resource record set from the hosted Azure DNS Zone if OffDelRR is set to 1. If OffDelRR is not set to 1, records are not deleted. The default value of OffDelRR is 0.

Monitor	Monitors if the resource record sets are present, and if all mappings are present in the hosted Azure DNS Zone. **Note:** If resource record sets are added or deleted manually, the agent does not monitor the change in state of these record sets.
Clean	Removes the resource record set from the hosted Azure DNS Zone if OffDelRR is set to 1. If OffDelRR is not set to 1, records are not deleted.

State definitions

ONLINE	Indicates that all the resource record sets are present in the Azure DNS Zone.
OFFLINE	Indicates that all or one of the resource record sets is not present in the Azure DNS Zone.
UNKNOWN	Indicates that a problem exists because of one of the following reasons: ■ Azure Python SDKs are not installed ■ Azure DNS Zone is not present ■ Roles are not configured properly ■ AzureDnsZoneResourceId is invalid ■ Resource Record Type (RSType) is not A, AAAA, CNAME, or PTR ■ ResRecord value does not have appropriate RSType value

Attributes

Table 3-18 Required attributes

Attribute	Description
AzureDnsZoneResourceId	Resource ID of the Azure DNS Zone. Type and dimension: string-scalar

Table 3-18 Required attributes *(continued)*

Attribute	Description
ResRecord	ResRecord is an association of DNS resource record values. Each ResRecord attribute consists of two values: DNS record key and DNS record data. Note that the record key must be a unique value. If the resource record list contains any invalid value as a part of the record key or a record data of the ResRecord attribute, the resource reports an UNKNOWN state. Resource records can be of the following types: ■ A ■ AAAA ■ CNAME ■ PTR Type and dimension: string-association
RSType	Record types supported by Azure DNS Zone. Type and dimension: string-scalar
AzureAuthResName	Name of the authentication agent resource that handles Azure related authentication. Type and dimension: string-scalar

Table 3-19 Optional attributes

Attribute	Description
TTL	Specifies the minimum time to live (in seconds) for all resource records. Default value is 3600 seconds. Type and dimension: int-scalar
OffDelRR	Determines if resource records will be deleted as part of the resource offline entry point. Default is false. Type and dimension: boolean-scalar

Resource type definition

```
type AzureDNSZone (
        static str ArgList[] = { AzureDnsZoneResourceId, RSType, ResRecords, TTL, OffDelRR,
        "AzureAuthResName:SubscriptionId", "AzureAuthResName:ClientId",
        "AzureAuthResName:SecretKey", "AzureAuthResName:TenantId" }
```

```
        str AzureDnsZoneResourceId
        str RSType
        str ResRecords{}
        int TTL = 3600
        boolean OffDelRR = 0
        str AzureAuthResName
)
```

Samples configurations

A

```
AzureDNSZone a_record (
AzureDnsZoneResourceId = "/subscriptions/6940a326-fgh6-40dd-b616-c1e9bbdf1d63/resourceGroups/
                         azureRG/providers/Microsoft.Network/dnszones/example.com
ResRecords = { "www" = "10.44.50.1,10.45.1.1", "vm1" = "10.44.50.2", "vm2" = "10.44.51.10",
              "@" = "10.45.55.66" }
RSType = "A"
TTL = 300
AzureAuthRes = "azure-auth-res"
)
```

AAAA

```
AzureDNSZone aaaa_record (
    AzureDnsZoneResourceId = "/subscriptions/6940a326-fgh6-40dd-b616-c1e9bbdf1d63/
      resourceGroups/resgrp/roviders/Microsoft.Network/dnszones/sampledomain.com"
    ResRecords = { "www" = " 2001:0db8:85a3:0000:0000:8a2e:0370:7334",
                   "vm1" = " 2607:f0d0:1002:0051:0000:0000:0000:0004",
                   "@" = "2001:0db8:85a3:0000:0000:8a2e:0370:7334" }
    RSType = "AAAA"
    TTL = 300
    AzureAuthRes = "azure-auth-res"
    )
```

CNAME

```
AzureDNSZone cname_record (
    AzureDnsZoneResourceId = "/subscriptions/6940a326-fgh6-40dd-b616-c1e9bbdf1d63/
       resourceGroups/resgrp/providers/Microsoft.Network/dnszones/sampledomain.com"
    ResRecords = { "vm1" = "vm1.alias.com", "vm2" = "vm2.alias.com" }
    RSType = "CNAME"
    TTL = 300
```

```
    AzureAuthRes = "azure-auth-res"
    )
```

PTR

```
AzureDNSZone ptr_record (
    AzureDnsZoneResourceId = "/subscriptions/6940a326-fgh6-40dd-b616-c1e9bbdf1d63/
       resourceGroups/resgrp/providers/Microsoft.Network/dnszones/sampledomain_ptr.com"
    ResRecords = { "11" = "vm1.alias.com", "222" = "vm2.alias.com" }
    RSType = "PTR"
    TTL = 300
    AzureAuthRes = "azure-auth-res"
    )
```

In this sample for PTR, 11 and 222 in ResRecords indicates the last block of the IP address. For example, 11 and 222 in 192.168.10.11 and 192.168.10.222 respectively.

Delegating a domain to Azure DNS

Delegate a domain to Azure DNS

1. Log on to the Azure portal.
2. Create a DNS zone.

   Azure DNS allocates name servers names for your zone.
3. In the **Favorites** pane, click **All resources** and get the name servers names from the DNS zone.
4. Using the DNS management tools of your existing registrar, edit the NS records and replace the NS records with the ones allocated by Azure DNS.

   Note: When delegating a domain to Azure DNS, you must use the name server names provided by Azure DNS. It is recommended to use all four name server names, regardless of the name of your domain. Domain delegation does not require the name server name to use the same top-level domain as your domain.
5. After completing the delegation, verify that the name resolution is working.

   If the delegation is set up correctly, the normal DNS resolution process finds the name servers automatically.
6. If the name resolution is not working, update the start of authority (SOA) record allocated by Azure DNS in your existing registrar.

Chapter 4

File share agents

This chapter includes the following topics:

About the file service agents

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

NFS agent

Starts and monitors the nfsd, mountd, statd, and lockd daemons required by all exported NFS file systems. Configure the NFS resource in a separate parallel service group with the AutoStart attribute set to 1.

You should configure only a single NFS resource in a service group on a node. If you have more than one service group that uses the NFS resource, the other service groups must use a Proxy resource. The Proxy resource can point to the NFS resource in the first group. Duplicate NFS resources will cause a problem when the NFS resources are brought online concurrently—only the NFS resource started

first will be successfully brought online, while the rest of the NFS resources may report online failure.

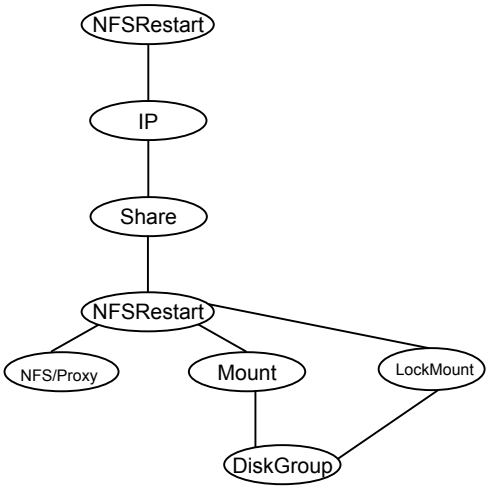
For important information about this agent,

See "Notes for NFS agent" on page 177.

Dependencies for NFS agent

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

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



Agent functions for NFS agent

Online	Starts NFS services on the system. For NFSv3, the services also include NSM (Network Status Monitor) and NLM (Network Lock Manager) service. If NFSv4 is enabled, it also starts NFSv4 name mapping daemon
Monitor	Monitors the NFS services running on the system. For NFSv3, it checks for version 2 and version 3 of NFS service along with other NFS services. For NFSv4, it also checks the availability of NFSv4 name mapping daemon.
Clean	Stops and restarts NFS services for all kernels. It also stops and restarts NSM and NLM services running on the system.

Attr_changed	When the Protocol attribute is changed, this function dynamically restarts the NFS services if the services are not running as per the protocol specified in the Protocol attribute.

State definitions for NFS agent

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

Attributes for NFS agent

Table 4-1 Optional attributes for Linux

Optional attributes	Description
GracePeriod	Required when the value of the NFSRestart attribute is 1. GracePeriod specifies the amount of time that lock recovery is allowed by the NFS server after its reboot. Type and dimension: integer-scalar Default: 90
LockFileTimeout	The NFS and the NFSRestart agents require a synchronization mechanism when the group to which they belong is in transition, for example going online or coming offline. A file serves as this synchronization mechanism. The LockFileTimeout attribute specifies the maximum time that the synchronization file exists. Type and dimension: integer-scalar Default: 180

Table 4-1 Optional attributes for Linux *(continued)*

Optional attributes	Description
Nproc	Specifies the number of concurrent NFS requests that the server can handle. Type and dimension: integer-scalar Default: 8 Example: 16
NFSSecurity	Specifies whether to start the NFS security daemon rpc.svcgssd or not. You must configure the type of security that NFS supports, for example: Kerberos. Type and dimension: boolean-scalar Default: 0
NFSv4Support	Specifies whether to start the NFSv4 daemon rpc.idmapd or not and whether to monitor nfsd version 4. Type and dimension: boolean-scalar Default: 0
Protocol	Specify the protocol to run the nfsd daemon. The following are the valid values: ■ tcp ■ udp ■ all The agent uses this attribute to ensure that the NFS daemon is running using the specified protocol. **Note:** VCS overrides any change to the NFS protocol outside of VCS control, so you must not change the NFS protocol outside of VCS control. Type and dimension: string-scalar Default: all Example: tcp

Table 4-1 Optional attributes for Linux *(continued)*

Optional attributes	Description
MountdOptions	Options for the mountd daemon. For more information, see the *mountd* manual page . For example: `-d all` In the above example, the agent executes the mountd daemon in verbose mode.
Port	Specifies the list of ports for NFS daemons. Valid values are: ■ NFSD *port_number* ■ STATD *port_number* ■ LOCKD *port_number* ■ MOUNTD *port_number* The NFS and NFSRestart agents use this attribute to ensure that the NFS daemons are running using the specified port. For the lockd daemon, you can specify different ports for TCP and UDP protocols. **Note:** VCS overrides any change to the NFS port outside of VCS control, so you must not change the NFS port outside of VCS control. Type and dimension: string-association Default: NULL Example 1: Port{} = { NFSD = 10000 } Example 2: Port{} = { NFSD = 10000, STATD = 100001, MOUNTD = 100002, LOCKD = 100003,100004 } In example 2, for the lockd daemon, port number 100003 is used for TCP and port number 100004 is used for UDP.

Resource type definition for NFS agent

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

```
    str MountdOptions
    str Protocol = all
    str Port{}
)
```

Notes for NFS agent

The NFS agent has the following notes:

- Prerequisites for NFS lock recovery
- Using NFSv4

Prerequisites for NFS lock recovery

If you plan on using lock recovery on a Linux system, store locking information on shared storage so that it is accessible to the system where NFS fails over. Using this information, NFS carries out lock recovery.

For more information, refer to the NFSRestart agent.

Using NFSv4

The NFS agent provides NFSv4 support to export shares using the attribute NFSv4Support. Only one of the Share resources that depends on the NFS resource needs a value of fsid=0 in its Options attribute. The shared directory that has the fsid=0 option becomes the root of all exports. The client needs to mount only this root file system instead of mounting all shares individually.

The syntax is:

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

Always use a slash (/) to end the path after the colon (:).

All the file systems, other than the root file system, needs to have the nohide option set in Options attribute of share resources. Set the nohide option so that authentic clients can seamlessly move through the tree of exported file systems just by mounting the root file system.

To enable NFSv4 support on your node, you must have the rpc_pipefs (pipe file system) mounted on the node. At boot time, rpc_pipefs is mounted on every Linux node. In situations where it is not mounted, mount rpc_pipefs on the cluster node.

To mount rpc_pipefs

- At the prompt on the node, enter the following:

```
# mount -t rpc_pipefs rpc_pipefs /var/lib/nfs/rpc_pipefs
```

Sample configurations for NFS agent

On each node in your cluster, you can find sample NFS, NFSRestart, and Share configurations in /etc/VRTSvcs/conf/sample_nfs/.

For more information regarding agent configuration, refer to the *Cluster Server Administrator's Guide*.

Debug log levels for NFS agent

The NFS agent uses the following debug log levels:

DBG_1, DBG_2, DBG_3, DBG_4, DBG_5

NFSRestart agent

The NFSRestart agent provides the following functionalities:

Manages NFS lock recovery service by recovering the NFS record locks after sudden server crash.

Prevents potential NFS ACK storms by terminating NFS server services before offline of NFS VIP to close all TCP connections with the NFS client.

If you have configured the NFSRestart agent for lock recovery, the NFSRestart agent starts the smsyncd daemon. The daemon copies the NFS locks from the local directory /var/lib/nfs to shared storage. The agent's online function copies the locks from shared storage to local directory /var/lib/nfs.

Note: On NFSv3 and NFSv4, the NFSRestart agent supports NFS lock recovery.

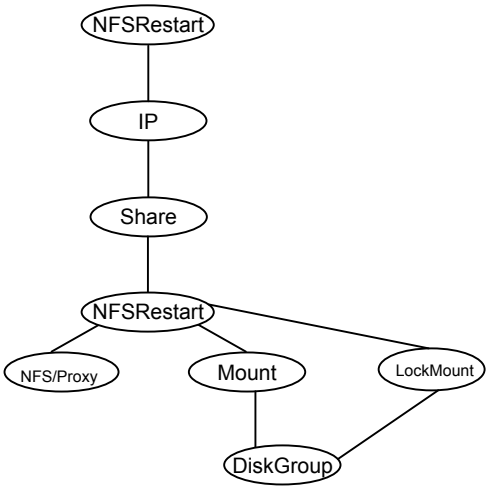
For important information about this agent, refer to Notes for NFSRestart agent

Dependencies for NFSRestart agent

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

You must use two NFSRestart resources in a service group. Both the NFSRestart resources provide combined protection from potential corruption of NFS locks and potential NFS ACK storms. The lower NFSRestart resource must have its Lower attribute set to 1. The upper NFSRestart resource should be at the top of the resource dependency tree and the lower NFSRestart resource should be below the Share resource in the resource dependency tree. The NFSRestart resources and the Share resources must be inside the same service group.

Figure 4-2 Sample service group that includes an NFSRestart resource



Agent functions for NFSRestart agent

The agent functions for this agent follow:

Online

For the lower NFSRestart resource:

- If the value of the NFSLockFailover attribute is 1, the agent terminates statd and lockd.
- If the value of the NFSLockFailover attribute is 1 and if NFSv4 is configured, the agent copies the NFSv4 state data of clients from the shared storage to local path.

For the upper NFSRestart resource:

- If the value of the NFSLockFailover attribute is 1, the agent copies the NFS record locks from shared storage to the /var/lib/nfs directory.
- Starts the statd and lockd daemons.
- Starts the smsyncd daemon to copy the contents of the /var/lib/nfs directory to the shared storage (LocksPathName) at regular two-second intervals.
- Starts the smsyncd daemon to copy the contents of the /var/statmon/sm directory to the shared storage (LocksPathName) and NFSv4 state data from local path to shared storage at regular two-second intervals.

Monitor	For the lower NFSRestart resource: ■ The monitor agent function does nothing. For the upper NFSRestart resource: ■ If the value of the NFSLockFailover attribute is 1, the agent monitors smsyncd daemon. It restarts the smsyncd daemon if it is not running.
Offline	For the lower NFSRestart resource: ■ Restarts all the NFS daemons that the upper NFSRestart resource stopped previously. For the upper NFSRestart resource: ■ Terminates the statd and lockd daemons to clear the lock state. ■ Terminates the nfsd and mountd daemons to close the TCP/IP connections. ■ Terminates the smsyncd daemon if the daemon is running.
Clean	For the lower NFSRestart resource: ■ Restarts all the NFS daemons that the upper NFSRestart resource stopped previously. For the upper NFSRestart resource: ■ Terminates the statd and lockd daemons to clear the lock state. ■ Terminates the nfsd and mountd daemons to close the TCP/IP connections. ■ Terminates the smsyncd daemon if the daemon is running.
Action	■ nfsconf.vfd Checks the runlevel information of the system service nfslock to confirm that the lock daemons do not come online automatically after reboot. ■ lockdir.vfd Verifies that the NFS lock directory (which is specified by the LocksPathName attribute of NFSRestart) is on shared storage.

State definitions

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

Attributes for NFSRestart agent

Table 4-2 Required attributes

Attribute	Description
NFSRes	Name of the NFS resource on the system. This attribute is required if the value of the NFSLockFailover attribute is 1. Type and dimension: string-scalar Example: "nfsres1"

Table 4-3 Optional attributes

Attribute	Description
LocksPathName	The path name of the directory to store the NFS lock information. This attribute is required when the value of the NFSLockFailover attribute is 1. The path that you specify for the LocksPathName attribute should be on shared storage. This is to ensure that it is accessible to all the systems where the NFSRestart resource fails over. Type and dimension: string-scalar Example: "/share1x"
NFSLockFailover	NFS Lock recovery is done for all the Share resources that are configured in the group of this resource. Type and dimension: boolean-scalar Default: 0
Lower	Defines the position of NFSRestart resource in the service group. The NFSRestart resource below the Share resource needs a value of 1. The NFSRestart resource on the top of the resource dependency tree has a Lower attribute value of 0. Type and dimension: integer-scalar Default: 0

Table 4-3 Optional attributes *(continued)*

Attribute	Description
LockFailoverAddress	Defines the virtual IP address from which notification should be sent to clients to reclaim locks on the NFSv3 file system. This address should be same as the address by which the clients access NFS shares. If the NFSLockFailover attribute value is 1, the upper NFSRestart resource requires this attribute. Type and dimension: string-scalar Examples: - "10.198.200.198" - "2001::7"

Resource type definition for NFSRestart agent

```
type NFSRestart (
    static keylist SupportedActions = { "lockdir.vfd",
     "nfsconf.vfd" }
    static str ArgList[] = { "NFSRes:Nproc",
     "NFSRes:GracePeriod", "NFSRes:NFSv4Support",
     NFSLockFailover, LocksPathName, Lower, State,
     "NFSRes:MountdOptions", "NFSRes:Protocol",
     "NFSRes:Port" }
    str NFSRes
    int Lower = 0
    str LocksPathName
    boolean NFSLockFailover = 0
)
```

Notes for NFSRestart agent

The NFSRestart agent has the following notes:

- About high availability fire drill
- Providing a fully qualified host name

About high availability fire drill

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

For NFSRestart resources, the high availability drill performs the following, it:

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

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

Providing a fully qualified host name

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

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

```
/etc/hosts
```

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

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

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

For example:

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

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

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

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

Sample configurations for NFSRestart agent

On each node in your cluster, you can find sample NFS, NFSRestart, and Share configurations in /etc/VRTSvcs/conf/sample_nfs/.

For more information regarding agent configuration, refer to the *Cluster Server Administrator's Guide*.

Basic agent configurations

For NFS lock recovery:

```
NFSRestart nfsrestart (
NFSRes = nfsres
LocksPathName="/shared_mnt/lockinfo"
NFSLockFailover = 1
Lower = 0
)
NFSRestart nfsrestart_L (
NFSRes = nfsres
LocksPathName="/shared_mnt/lockinfo"
NFSLockFailover = 1
Lower = 1
)
```

For no NFS lock recovery:

```
NFSRestart nfsrestart (
NFSRes = nfsres
)
NFSRestart nfsrestart_L (
NFSRes = nfsres
Lower = 1
)
```

Debug log levels for NFSRestart agent

The NFSRestart agent uses the following debug log levels:

DBG_1, DBG_3, DBG_4, DBG_5

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 exported are on shared disks.

For important information on this agent, refer to:

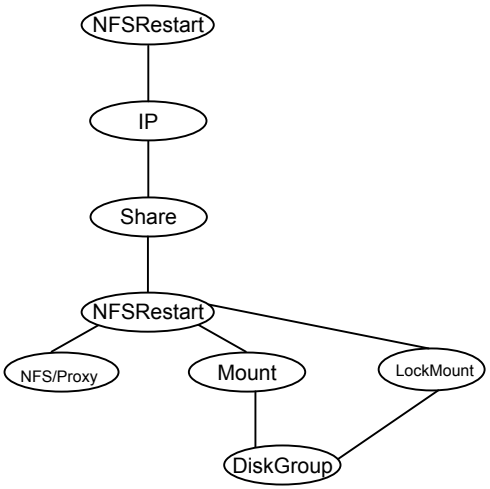
Notes for Share agent

Dependencies for Share agent

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

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

Figure 4-3 Sample service group that include a Share resource



Agent functions for Share agent

Online	Exports (shares) a directory to the specified client.
Offline	Unshares the exported directory from the client.
Monitor	Verifies that the shared directory is exported to the client.

Clean	Terminates all ongoing resource actions and takes the resource offline, forcibly when necessary.
Action	direxists.vfd Checks if the path specified by the PathName attribute exists on the cluster node. If the path name is not specified, it checks if a corresponding mount point is available to ensure that the path is on shared storage.

State definitions for Share agent

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

Attributes for Share agent

Table 4-4 Required attributes

Required attribute	Description
PathName	Pathname of the file system to be shared. Type and dimension: string-scalar Example: "/share1x"
NFSRes	This attribute has been deprecated.

Table 4-5 Optional attributes

Optional attribute	Description
Client	The Share agent accepts as many clients as the user wishes provided all the clients are exported the same 'PathName'. Client or host where the directory specified by PathName is exported. The client can be a wild card (*), a fully qualified domain name (FQDN) including the host name, or an IP address. For more information on specifying IP address, refer to About entering an IP address in the Client attribute. Type and dimension: string-scalar Example: If "outland" is the host name, the FQDN hostname is outland.example.com.
Options	Options to the `exportfs` command. When specifying multiple options, separate them with commas, for example: "rw, no_root_squash" For more information about the `exportfs` command and its options, refer to the exportfs manual page. Type and dimension: string-scalar Default = "ro, async, wdelay, root_squash"
OtherClients	The Client attribute can be assigned one FQDN host name or IP address, whereas multiple FQDN host names, or IP addresses can be assigned to the **OtherClients** field. For more information on specifying IP address, refer to About entering an IP address in the Client attribute. A combination of 'Client' and 'OtherClients' can be used to specify the host names. If both of the Client and OtherClients attributes are left unspecified, the PathName is exported to the world (*). Type and dimension: string-vector

Resource type definition for Share agent

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

```
str OtherClients[]
str Options
str NFSRes
)
```

Notes for Share agent

The following section contains notes on the Share agent.

- Support for spaces in directory names
- High availability fire drill
- About entering an IP address in the Client attribute

Support for spaces in directory names

The Share agent supports directory names with spaces. The space can be leading, trailing, or in the middle of the name. If the directory name has a trailing space, provide an extra "/" at the end of the PathName attribute of a Share resource. Note that the agent does not support spaces created using the TAB key.

High availability fire drill

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

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

About entering an IP address in the Client attribute

You can enter an IP address as a part of the Client or OtherClients attribute. If you specify an IP address, use the form `a.b.c.d/32`. Using this form of IP address prevents the monitor entry point from reporting a Share resource as offline even if the sharetab entry is present. This occurs on some SLES systems where the `exportfs` command tries to resolve the client IP address to the host name and rewrite the sharetab entry with the host name. If you specify the IP address as `a.b.c.d/32`, `exportfs` does not recognize this as an IP address and does not attempt to resolve the address to a host name.

Sample configurations for Share agent

On each node in your cluster, you can find sample NFS, NFSRestart, and Share configurations in /etc/VRTSvcs/conf/sample_nfs/.

For more information regarding agent configuration, refer to the *Cluster Server Administrator's Guide*.

Debug log levels for Share agent

The Share agent uses the following debug log levels:

DBG_1, DBG_3, DBG_4, DBG_5

About the Samba agents

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

- Filespace
- Printer
- WINS
- Domain Master

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

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

The Samba agents

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

Before using the Samba agents

- Verify that smbd and nmbd always run as daemons. Verify that they cannot be started using the meta-daemon inetd.
- Verify that the smbd and nmbd daemons are in the path environment variable. The default path of the smbd and nmbd daemons is: /usr/sbin

For more information on configuring these paths, refer to the description of the SambaTopDir attribute.

- Verify that Samba is configured properly and that the Samba configuration file is identical on all cluster systems. The user can replicate the file or store it on a shared disk accessible from all cluster systems.
- If configuring Samba as a WINS server or Domain Master, verify that the Samba lock directory is on the shared disk. This ensures that the WINS server database and Domain Master are created on the shared disk.

Supported versions for Samba agents

VCS Samba suite of agents support Samba version 3.0 and above. Please check your samba version using the following command:

```
# smbd -V
```

Notes for configuring the Samba agents

The following notes describe configuration considerations for the Samba agents.

Enabling VCS to detect services started and stopped by smb

Edit the samba file to enable VCS to detect the Samba services that are started and stopped using the following commands:

- `/etc/init.d/smb start` or
  `service smb start`
- `/etc/init.d/smb stop` or
  `service smb stop`

Edit the /etc/sysconfig/samba file. Add *-s smb.conf-filepath* to SMBDOPTIONS. This change is required because the SambaServer agent monitors the resource by matching the configuration file present in running process's arguments list with the value configured in ConfFile attribute.

For example:

```
$ cat /etc/sysconfig/samba
# Options to smbd
SMBDOPTIONS="-D -s /etc/samba/smb.conf"
# Options to nmbd
NMBDOPTIONS="-D -s /etc/samba/smb.conf"
```

```
# Options for winbindd
WINBINDOPTIONS=""
```

Configuring multiple SambaServer resources

For configuring multiple SambaServer resources, configure the SocketAddress attribute with the unique value of the address where the respective samba daemon listens for connections. Configure the SambaServer resource as a parent resource of the IP resource. Configure this IP resource with the SocketAddress attribute value.

Configuring Samba for non-standard configuration files or non-standard lock directories

Configure the PidFile attribute if you use a non-standard configuration file for Samba or if the lock directory (the directory where Samba pid file resides) for Samba is different than the default location. Use the following command to check the standard locations for the Samba configuration file and the lock directory:

To check for the default value of the Samba configuration file

- Enter the following command:

  ```
  # smbd -b | grep CONFIGFILE
  ```

To check for the default location of the Samba pidfile

- Enter the following command:

  ```
  # smbd -b | grep PIDDIR
  ```

SambaServer agent

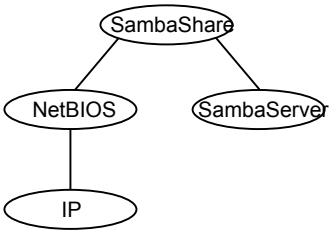
The SambaServer agent starts, stops, and monitors the smbd process as a daemon. Only one resource of this type is permitted. You can use the agent to make a smbd daemon highly available.

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

Dependencies for SambaServer agent

No dependencies exist for the SambaServer resource.

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



Agent functions for SambaServer agent

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

State definitions for SambaServer agent

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

Attributes for SambaServer agent

Table 4-6 Required attributes

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

Table 4-7 Optional attributes

Optional attribute	Description
IndepthMonitorCyclePeriod	Number of monitor cycles after which the in-depth monitoring is performed. For example, the value 5 indicates that the agent monitors the resource in-depth every five monitor cycles. The value 0 indicates that the agent will not perform in-depth monitoring for the resource. Type and dimension: integer-scalar Default: 5
Ports	Ports where Samba accepts connections. To run Samba over NBT (NetBios over TCP/IP), set this attribute to 139. To run Samba directly over TCP/IP, set this attribute to 445. Type and dimension: integer-vector Default: 139, 445
ResponseTimeout	Number of seconds the agent waits to receive the session response packet after sending the session request packet. For example, the value 5 indicates that the agent waits for five seconds before receiving the session response packet. Configure this attribute if in-depth monitoring is enabled. Type and dimension: integer-scalar Default: 10

Table 4-7 Optional attributes *(continued)*

Optional attribute	Description
PidFile	The absolute path to the Samba daemon pid file. This file contains the process ID of the monitored smbd process. Configure this attribute if you are using a non-standard configuration file name or path. If this attribute is not configured for non-standard configuration file names, the agent checks the smbd-*ConfFile*.pid file for monitoring the resource. Type and dimension: string-scalar Example: "/var/run/smbd.pid"

SambaTopDir	Parent path of Samba daemon and binaries.
	Type and dimension: string-scalar
	Example: "/usr"
	SambaServer agent uses SambaTopDir attribute value in an open entry point to determine the complete path of samba executables. If this attribute is configured after the resource is enabled, please disable and enable the resource again to bring this into effect as follows:
	`# hares -modify <res> Enabled 0`
	`# hares -modify <res> Enabled 1`

SocketAddress	The IP address where the Samba daemon (smbd) listens for connections. Configure the SocketAddress attribute if you are configuring multiple SambaServer resources on a node. Type and Dimension: string-scalar Example: “10.128.10.14”, "2001::10"

Resource type definitions for SambaServer agent

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

```
str SocketAddress
)
```

Sample configurations for SambaServer agent

The sample configurations for this agent follow:

```
SambaServer samba_server (
ConfFile = "/etc/samba/smb.conf"
LockDir = "/usr/lock/samba"
IndepthMonitorCyclePeriod = 3
ResponseTimeout = 15
)
```

Debug log levels for SambaServer agent

The SambaServer agent uses the following debug log levels:

DBG_1, DBG_2, DBG_3, DBG_4, DBG_5

SambaShare agent

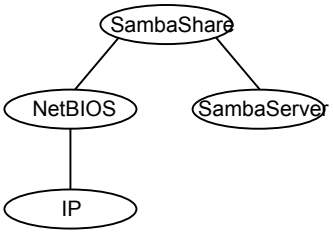
The SambaShare agent adds, removes, and monitors a share by modifying the specified Samba configuration file. You can use the agent to make a Samba Share highly available.

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

Dependencies for SambaShare agent

SambaShare resources depend on the SambaServer, NetBios and Mount resources.

Figure 4-5 Sample service group for a SambaShare resource



Agent functions for SambaShare agent

Online	Edits the samba configuration file and adds the shares.
Offline	Removes the shares from the configuration file.
Monitor	Issues the command `smbclient` to check if the specified shares exist.
Clean	Terminates all ongoing connections with the particular samba share, removes its entry from the samba configuration file and reloads the configuration.

State definitions for SambaShare agent

ONLINE	Indicates that the share is available.
OFFLINE	Indicates that the share is not available.
FAULTED	Indicates that the share has become unavailable outside of VCS control.
UNKNOWN	Indicates that the agent could not determine the state of the resource.

Attributes for SambaShare agent

Table 4-8 Required attributes

Required attribute	Description
SambaServerRes	Name of the SambaServer resource. Type and dimension: string-scalar Example: "smb_res1"
ShareName	Name of the share resource as exported by samba. **Note:** This name can be different from the SambaShare resource name. Type and dimension: string-scalar Example: "share1"

Table 4-8 Required attributes *(continued)*

Required attribute	Description
ShareOptions	List of parameters for the share attributes. These parameters are specified as name=value pairs, with each pair separated by a semicolon (;). Type and dimension: string-scalar Example: "path=/shared; public=yes; writable=yes"

Resource type definition for SambaShare agent

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

Sample configuration for SambaShare agent

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

Debug log levels for SambaShare agent

The SambaShare agent uses the following debug log levels:

DBG_1, DBG_3, DBG_5

NetBios agent

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

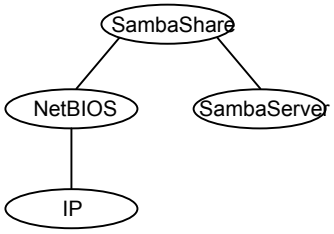
The agent sets, monitors, and resets the names and network interfaces by which the Samba server is known. The agent also sets, monitors and resets Samba to act as a WINS server or domain master or both.

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

Dependencies for NetBios agent

The NetBios resource depends on the IP or the IPMultiNIC resource if the virtual IP address configured in the IP/IPMultiNIC resource is being used in the Interfaces attribute of the NetBios resource.

Figure 4-6 Sample service group that includes a NetBIOS resource



Agent functions for NetBios agent

Online	Updates the Samba configuration with the NetBIOS name, all NetBIOS aliases and network interfaces, WINS support, and domain master options specified in the NetBIOS resource. Starts the nmbd daemon.
Offline	Removes the NetBIOS name, all NetBIOS aliases and network interfaces, WINS support, and domain master options specified in the NetBIOS resource from the Samba configuration file. Stops the nmbd daemon.
Monitor	Verifies that the Samba configuration contains the NetBIOS name, all NetBIOS aliases and network interfaces, WINS support, and domain master options specified in the NetBIOS resource. Also verifies that the nmbd daemon is running by reading its pid file.

Clean	Removes the NetBIOS name, all NetBIOS aliases and network interfaces, WINS support, and domain master options specified in the NetBIOS resource from the Samba configuration file. Stops the nmbd daemon, forcibly when necessary.

State definitions for NetBios agent

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

Attributes for NetBios agent

Table 4-9 Required attributes

Required attribute	Description
NetBiosName	Name by which the Samba server is known in the network. Type and dimension: string-scalar Example: "samba_demon" **Note:** Samba has a limitation of 15 characters for NetBios names and aliases.
SambaServerRes	Name of the SambaServer resource. Type and dimension: string-scalar Example: "smb_res1"

Table 4-10 Optional attributes

Optional attribute	Description
Interfaces	List of network interfaces on which Samba handles browsing. Type and dimension: string-vector Example: "172.29.9.24/16" **Note:** If you have configured the SocketAddress attribute value for the corresponding SambaServer resource, then you must also configure the same value paired with the appropriate netmask in the list of interfaces.
NetBiosAliases	List of additional names by which the Samba server is known in the network. Type and dimension: string-vector Example: { host1_samba, myname } **Note:** Samba has a limitation of 15 characters for NetBios names and aliases.
WinsSupport	If set to 1, this flag causes the agent to configure Samba as a WINS server. Type and dimension: integer-scalar Default: 0
DomainMaster	If set to 1, the agent sets Samba as Domain Master. Note that there can be only one domain master in a domain. Type and dimension: integer-scalar Default: 0
PidFile	The absolute path to the NetBIOS daemon pid file. This file contains the process ID of the monitored nmbd process. Configure this attribute if you are using a nonstandard configuration file name or path. If this attribute is not configured for non-standard configuration file names, the agent checks for the nmbd-*ConfFile*.pid file for resource monitoring. Type and dimension: string-scalar Example: "/var/run/nmbd.pid"

Resource type definition for NetBios agent

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

Sample configuration for NetBios agent

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

Debug log levels for NetBios agent

The NetBios agent uses the following debug log levels:

DBG_1, DBG_5

Chapter 5

Service and application agents

This chapter includes the following topics:

About the services and applications agents

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

Apache HTTP server agent

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

This agent supports Apache HTTP server 2.0, 2.2, and 2.4. It also supports IBM HTTP Server 1.3, 2.0, and 7.x.

You can view the latest support information for Apache HTTP server at: https://sort.veritas.com/agents

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

This agent is IMF-aware and uses the AMF kernel driver for IMF notification.

IMF support is enabled by default. In VCS 6.1 and later, only PRON IMF monitoring is supported and the IMF Mode attribute value is set to 2.

For more information about IMF and intelligent resource monitoring, refer to the *Cluster Server Administrator's Guide*.

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

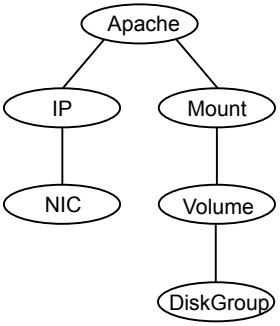
The agent performs detailed monitoring on the Apache resource. You can tune the frequency of detailed monitoring with the LevelTwoMonitorFreq attribute. By default, the agent monitors the Apache resource after every fifth monitor cycle to confirm the health of the resource.

See "Apache HTTP server notes" on page 210. for more information regarding this agent.

Dependencies

This type of resource depends on IP and Mount resources.

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



Agent functions

Online	To start the Apache HTTP server, the agent: ■ Executes the httpdDir/httpd program with the appropriate arguments if the httpdDir program specifies the full path of the directory in which the httpd binary file is located. ■ Alternatively, if the httpdDir attribute specifies the full path of the Apache HTTP server binary file, the binary file is executed with appropriate arguments. When you specify a file with the EnvFile attribute, the file is sourced before the agent executes the Apache HTTP server commands.
Offline	To stop the Apache HTTP server, the agent: ■ Executes the httpdDir/httpd program with the appropriate arguments, if httpdDir specifies the full path of the directory in which the httpd binary file is located. ■ Alternatively, if the httpdDir attribute is used to specify the full path of the Apache HTTP server binary, the binary file is executed with appropriate arguments. ■ Sends a TERM signal to the HTTP Server parent process (Apache). When you specify a file with the EnvFile attribute, the file is sourced before the agent executes the Apache HTTP server commands.
Monitor	Monitors the state of the Apache server. First it checks for the processes, next it can perform an optional state check.
Clean	Removes the Apache HTTP server system resources that might remain after a server fault or after an unsuccessful attempt to online or offline. These resources include the parent httpd daemon and its child daemons.
Action	checkconffile.vfd Checks for the existence of the Apache configuration file and the existence of the directory that contains the httpd binary that is used during start up. For a local installation, if the config file or HttpdDir is not found, make sure that it exists on the failover node.
imf_init	Initializes the agent to interface with the AMF kernel driver. This function runs when the agent starts.

imf_getnotification	Gets notification about resource state changes during the online operation. This function runs after the agent initializes with the AMF kernel driver. The agent continuously waits for notification and takes action on the resource upon notification.
imf_register	Registers the resource entities for online monitoring with the AMF kernel driver. The Apache agent reports the resource as online when the parent Apache HTTP server process and at least one child HTTP server process is running. The Process ID of the parent Apache HTTP server process and one child process found on the system is registered with AMF. For example, the function registers the PID of the process that requires online monitoring. This function runs for each resource after the resource goes into steady online state.

State definitions

ONLINE	Indicates that the Apache server is running.
OFFLINE	Indicates that the Apache server is not running. Can also indicate that the administrator has stopped the HTTP server gracefully. Note that the agent uses the PidFile attribute for intentional offline detection.
UNKNOWN	Indicates that a problem exists with the configuration.
FAULTED	Indicates that the Apache server has stopped unexpectedly or is not responding (if in-depth monitoring is enabled) outside of VCS control.

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: "/etc/httpd/conf/httpd.conf"

Table 5-1 Required attributes *(continued)*

Required attribute	Description
httpdDir	Full path of the Apache HTTP server binary file or full path of the directory in which the httpd binary file is located. Type and dimension: string-scalar Example: "/usr/apache2/bin"
PidFile	This attribute is required when you want to enable the detection of a graceful shutdown outside of VCS control. See Table 5-2 on page 206.

Table 5-2 Optional attributes

Optional attribute	Description
DirectiveAfter	A list of directives that httpd processes after reading the configuration file. Type and dimension: string-association Example: DirectiveAfter{} = { KeepAlive=On }
DirectiveBefore	A list of directives that httpd processes before it reads the configuration file. Type and dimension: string-association Example: DirectiveBefore{} = { User=nobody, Group=nobody }
User	Account name the agent uses to execute the httpd program. If you do not specify this value, the agent executes httpd as the root user. Type and dimension: string-scalar Example: "apache1"

Table 5-2 Optional attributes *(continued)*

Optional attribute	Description
EnableSSL	If this attribute is set to 1 (true) the online agent function will add support for SSL, by including the option `-DSSL` in the start command. For example: `/usr/sbin/httpd -f path_to_httpd.conf -k start -DSSL` Where path_to_httpd.conf file is the path to the httpd.conf file. If this attribute is set to 0 (false) the agent excludes the SSL support. Type and dimension: boolean-scalar Default: 0 Example: "1"
HostName	The virtual host name that is assigned to the Apache server instance. The host name is used in second-level monitoring for benchmarking the Apache HTTP server. You can use IPv4 or IPv6 addresses for the HostName attribute. **Note:** The HostName attribute is required only if you enable in-depth monitoring by setting the LevelTwoMonitorFreq attribute. Type and dimension: string-scalar Example: "web1.example.com"
Port	Port number where the Apache HTTP server instance listens. The port number is used in second-level monitoring for benchmarking the Apache HTTP server. Specify this attribute only if you have enabled in-depth monitoring by setting the LevelTwoMonitorFreq attribute. Type and dimension: integer-scalar Default: 80 Example: "80"
EnvFile	Full path and file name of the file that is sourced before executing Apache HTTP server commands. Specifying this attribute is optional. If EnvFile is specified, the shell for the user must be Bourne, Korn, or C shell. Type and dimension: string-scalar Example: "/apache/server1/bin/envvars"

Table 5-2 Optional attributes *(continued)*

Optional attribute	Description
PidFile	The PidFile attribute sets the file to which the server records the process ID of the daemon. The value of PidFile attribute must be the absolute path where the Apache instance records the PID. This attribute is required when you want the agent to detect the graceful shutdown of the Apache HTTP server. For the agent to detect the graceful shutdown of the Apache HTTP server, the value of the IntentionalOffline resource type attribute must be 1 (true). Type and dimension: string-scalar Example: /var/run/httpd.pid
SharedObjDir	Full path of the directory in which the Apache HTTP shared object files are located. Specifying this attribute is optional. It is used when the HTTP Server is compiled using the SHARED_CORE rule. If you specify this attribute, the directory is passed to the `-R` option when executing the httpd program. Refer to the httpd man pages for more information about the `-R` option. Type and dimension: boolean-scalar Example: "/apache/server1/libexec"
ResLogLevel	This attribute has been deprecated. Use the resource type attribute LogDbg to enable debug logs. Set LogDbg attribute to DBG_5 to enable debug logs for the Apache HTTP server agent. By default, setting the LogDbg attribute to DBG_5 enables debug logs for all Apache resources in the cluster. If debug logs must be enabled for a specific Apache resource, override the LogDbg attribute. For information on how to use the LogDbg attribute, refer to the *Cluster Server Administrator's Guide*.

Table 5-2 Optional attributes *(continued)*

Optional attribute	Description
LevelTwoMonitorFreq	Specifies the frequency at which the agent must perform second-level or detailed monitoring. You can also override the value of this attribute at the resource level. The value indicates the number of monitor cycles after which the agent will monitor Apache in detail. For example, the value 5 indicates that the agent will monitor Apache in detail after every five online monitor intervals. Type and dimension: integer-scalar Default: 0

Table 5-3 Resource type attribute

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

Resource type definition

```
type Apache (
    static keylist SupportedActions = { "checkconffile.vfd" }
    static str ArgList[] = { ResLogLevel, State, IState, httpdDir,
    SharedObjDir, EnvFile, PidFile, HostName, Port, User,
    ConfigFile, EnableSSL, DirectiveAfter, DirectiveBefore }
    str ResLogLevel = INFO
    str httpdDir
    str SharedObjDir
    str EnvFile
    str PidFile
    str HostName
    int Port = 80
    str User
    str ConfigFile
    str DirectiveAfter{}
    str DirectiveBefore{}
    boolean EnableSSL
```

```
    static int IMF{} = { Mode = 2, MonitorFreq = 5, RegisterRetryLimit = 3 }
    static str IMFRegList[] = { ConfigFile, httpdDir }

    static boolean IntentionalOffline = 0
)
```

Apache HTTP server notes

The Apache HTTP server has the following notes:

- Tasks to perform before you use the Apache HTTP server agent
- About detecting application failure
- About bringing an Apache HTTP server online outside of VCS control
- About high Availability fire drill
- Using Apache agent with IMF

Tasks to perform before you use the Apache HTTP server agent

Before you use this agent, perform the following tasks:

- Install the Apache server on shared or local disks.
- Ensure that you are able to start the Apache HTTP server outside of VCS control, with the specified parameters in the Apache configuration file (for example: /etc/apache/httpd.conf). For more information on how to start the server:
  See "About bringing an Apache HTTP server online outside of VCS control" on page 211.
- Specify the location of the error log file in the Apache configuration file for your convenience (for example: ErrorLog /var/apache/logs/error_log).
- Verify that the floating IP has the same subnet as the cluster systems.
- If you use a port other than the default 80, assign an exclusive port for the Apache server.
- Verify that the Apache server configuration files are identical on all cluster systems.
- Verify that the Apache server does not autostart on system startup.
- Verify that `inetd` does not invoke the Apache server.
- The service group has disk and network resources to support the Apache server resource.
- Assign a virtual host name and port to the Apache server.

- Verify that you are able to start the Apache HTTP server outside of VCS control in non-interactive manner. For example, the `startup` command should not prompt for any password or any other interactive questions.

About detecting application failure

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

The first check determines the state of the Apache HTTP server. The check determines the state by searching for the existence of the parent httpd daemon. It also searches for at least one child httpd daemon. If the parent process and at least one child do not exist, VCS reports the resource as offline. If they do exist and if the agent attribute LevelTwoMonitorFreq is set, the Apache agent uses the Apache Benchmarking utility "ab" to perform detail monitoring. If the exit code of the "ab" utility is 0 and if the command output contains "Benchmarking HostName", the agent considers the server online, else the agent considers the server offline.

If the binary file `ab` is not found, Apache agent uses the `ab2` binary file for detail monitoring.

About bringing an Apache HTTP server online outside of VCS control

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

To start an Apache HTTP server outside of VCS control

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

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

   In the above-mentioned command, replace *httpdDir* with /apache/v2.2/bin and *ConfigFile* with /apache/v2.2/conf/httpd.conf. When fully formed, the start example looks like:

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

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

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

   Note: You can specify the full path of a binary file without having httpd as part of httpdDir attribute.

   For example: `/usr/sbin/apache2 -f /etc/httpd/conf/httpd.conf -k start`

About high Availability fire drill

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

For Apache resources, when the Apache HTTP server is installed locally, the high availability fire drill checks for the validity of these attributes:

- ConfigFile
- httpdDir

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

Using Apache agent with IMF

The Apache agent only supports intelligent monitoring during the online operation. The agent registers the following two processes for Apache IMF:

- Process with parent PID init.
- Child process with the maximum elapsed time.

By default, the IMF Mode is set to 2. If the IMF Mode is set to 1 or 3, the offline registration with IMF fails repeatedly until RegisterRetryLimit is reached.

Sample configurations

Running two instances of httpd for Linux

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

Each Apache resource requires an online IP resource. In this example, each Apache resource requires an online mount resource to mount block devices from disks:

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

Apache myapacheWeb (
httpdDir = "/mnt/apache/bin"
ConfigFile = "/mnt/apache/conf/httpd.conf"
HostName = "server1.example.com"
Port = 80
)

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

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

```
)

Mount myapacheMnt(
MountPoint="/mnt/apache/"
BlockDevice="/dev/sdd2"
FSType = ext3
FsckOpt = "-y"
)

myapacheIP requires myapacheNIC
myapacheWeb requires myapacheIP
myapacheWeb requires myapacheMnt

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

Apache myapacheWeb2(
httpdDir = "/mnt/apache1/bin"
ConfigFile = "/mnt/apache1/conf/httpd.conf"
HostName = "server2.example.com"
Port = 8080
)

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

NIC myapacheNIC2(
Device="eth1"
)

Mount myapacheMnt2(
MountPoint="/mnt/apache1/"
BlockDevice="/dev/sdc3"
FSType = ext3
FsckOpt = "-y"
)

myapacheIP2 requires myapacheNIC2
```

```
myapacheWeb2 requires myapacheIP2
myapacheWeb2 requires myapacheMnt2
```

Sample main.cf file

A sample main.cf file follows:

```
include "types.cf"

cluster Cluster1 (
    UserNames = { admin = xxxxxx }
)

system SystemA (
)
system SystemB (
)

group Web1 (
    SystemList = { SystemA = 0, SystemB = 1 }
)

    DiskGroup Web1_dg (
        DiskGroup = web1
    )

    Volume Web1_vol (
        DiskGroup = web1
        Volume = volweb1
    )

    IP Web1_ip (
        Device = eth0
        Address = "10.212.88.220"
        NetMask = "255.255.254.0"
    )

    Mount Web1_mnt (
        MountPoint = "/apache/srvr01"
        BlockDevice = "/dev/vx/dsk/web1/volweb1"
        FSType = vxfs
```

```
        FsckOpt = "-y"
    )

    NIC Web1_nic (
        Device = eth0
    )

    Apache Web1_http (
        HostName = spartan
        Port = 80
        httpdDir = "/apache/srvr01/bin"
        EnvFile = "/apache/srvr01/bin/envvars"
        PidFile = /apache/srvr01/log/httpd.pid"
        ConfigFile = "/apache/srvr01/conf/httpd.conf"
        IntentionalOffline = 1
    )

    Web1_ip requires Web1_nic
    Web1_mnt requires Web1_vol
    Web1_vol requires Web1_dg
    Web1_http requires Web1_ip
    Web1_http requires Web1_mnt
```

Basic IPv6 configuration

The following is a basic IPv6 configuration for the resource.

```
group ipv6group (
    SystemList = { sysA = 0, sysB = 1 }
    )

    Apache ipv6group_apache_res (
        HostName = "fd4b:454e:205a:110:211:25ff:fe7e:118"
        PidFile = "/myapache/apache/logs/httpd.pid"
        httpdDir = "/myapache/apache/bin"
        ConfigFile = "/myapache/apache/conf/httpd.conf"
        IntentionalOffline = 1
        )

    DiskGroup ipv6group_dg_res (
        DiskGroup = dg01
        )

    IP ipv6group_ip_res (
```

```
        Device = eth0

        Address = "fd4b:454e:205a:110:211:25ff:fe7e:118"
        PrefixLen = 64
        )

    Mount ipv6group_mnt_res (
        MountOpt = rw
        FsckOpt = "-n"
        BlockDevice = "/dev/vx/dsk/dg01/vol01"
        MountPoint = "/myapache/apache"
        FSType = vxfs
        )

    NIC ipv6group_nic_res (

Device = eth0


)
Volume ipv6group_vol_res (
        Volume = vol01
        DiskGroup = dg01
        )

    ipv6group_apache_res requires ipv6group_mnt_res
    ipv6group_apache_res requires ipv6group_ip_res
    ipv6group_mnt_res requires ipv6group_vol_res
    ipv6group_vol_res requires ipv6group_dg_res
    ipv6group_ip_res requires ipv6group_nic_res
```

Sample output of the amfstat command

The following is a sample output of the `amfstat` command:

```
IMFD
=====
RID     PID
7       7929886

Registered Reapers (3):
=======================
RID PID       MONITOR TRIGG   REAPER
29  13041840  1       0       VCSMountAgent
30  9175060   2       0       Apache
```

```
31  12189854  1       0       DiskGroup

Process ONLINE Monitors (2):
============================
RID   R_RID   PID       GROUP
34    30      6488150   httpd_server
35    30      8847606   httpd_server

Mount ONLINE Monitors (1):
==========================
RID R_RID FSTYPE DEVICE          MOUNTPOINT
33  29    vxfs   /dev/vx/dsk/Apache_Conf/apache_vol /Apache
GROUP   CONTAINER
Apache_mnt none

DG online Monitors (1):
=========================
RID    R_RID   GROUP           DGName
32     31      Apache_dg       Apache_Conf
```

Debug log level

The Apache agent uses the following debug log level:

DBG_5

Application agent

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

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

You can monitor the application in the following ways:

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

- Any combination of the above

Prevention Of Concurrency Violation (ProPCV) can be enabled to prevent an online resource on a node from coming online on another node, outside of VCS control, in the same cluster. In that, ProPCV prevents the execution of StartProgram and processes that are configured in MonitorProcesses on the offline node. This action prevents data corruption of resources and detects concurrency violation at an early stage. The attribute can only be set for a local failover type group. To enable this feature you need to set the ProPCV attribute value to 1. For more information about ProPCV, refer to the *Cluster Server Administrator's Guide.*

The Application agent also lets you start or stop an application without monitoring it continuously. Use the StartOnly attribute of this agent to indicate that VCS should only start or stop the application and not perform a monitor operation. When StartOnly is set to 1, the agent uses the return values of StartProgram and StopProgram to determine whether to report the ONLINE or the OFFLINE state for the resource.

Limitations of the start-only option for an application:

- If the application does not come online after multiple attempts—determined by OnlineRetryLimit, the agent reports the state as OFFLINE|FAULTED.
- If the application does not go offline, the agent reports the state as ONLINE|UNABLE TO OFFLINE.

IMF awareness

The Application agent is IMF-aware and uses asynchronous monitoring framework (AMF) kernel driver for IMF notification. For more information about IMF and intelligent resource monitoring, refer to the *Cluster Server Administrator's Guide*.

For more information about IMF-related Application agent functions, see Agent functions.

High availability fire drill for Application agent

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

- The availability of the specified program and execution permissions for the specified program (program.vfd)
- The existence of the specified user on the host (user.vfd)
- The existence of the same binary on all nodes (cksum.vfd)

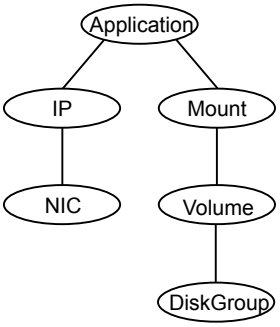
For more information, refer to the *Cluster Server Administrator's Guide*.

Dependencies for Application agent

No fixed dependency exists for Application agent.

Depending on how you plan to use it, an Application type of resource can depend on IP and Mount resources. Alternatively, instead of the IP resource you can also use the IPMultiNIC resource.

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



Agent functions

Online

Runs the executable file with the parameters that are specified in the StartProgram attribute in the context of the specified user.

To bring the resource online, the agent function performs the command:

```
su [-] user -c executable_to_online_resource
```

Offline

Runs the executable file with the parameters that are specified in the StopProgram attribute in the context of the specified user.

To take the resource offline, the agent function performs the command:

```
su [-] user -c executable_to_offline_resource
```

Monitor	How the agent performs this function depends on the attributes that you specify: ■ If you specify the MonitorProgram attribute, the agent executes the user-defined monitor program in the user-specified context. ■ If you specify the PidFiles attribute, the routine verifies that the process ID that is found in each listed file is running. ■ If you specify the MonitorProcesses attribute, the routine verifies that each listed process is running in the context of the user that you specify. You can use any combination of the MonitorProgram, PidFiles, or MonitorProcesses attributes to monitor the application. If this function determines that one or more of the processes that are specified in either PidFiles or MonitorProcesses are not running, it returns OFFLINE. If the process terminates ungracefully, the monitor returns OFFLINE, and a failover occurs. To monitor the resource, the agent function performs the command: `su [-] user -c executable_to_monitor_resource`
imf_init	Initializes the agent to interface with the asynchronous monitoring framework (AMF) kernel driver. This function runs when the agent starts up.
imf_getnotification	Gets notification about resource state changes. This function runs after the agent initializes with the AMF kernel driver. The agent continuously waits for notification and takes action on the resource upon notification.
imf_register	Registers the resource entities, which the agent must monitor, with the AMF kernel driver. For example, the function registers the PID for online monitoring of a process. This function runs for each resource after the resource goes into steady state (online or offline). The Application agent uses IMF for the processes configured with PidFiles and the MonitorProcesses attribute.

Clean

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

To forcefully stop the resource, the agent function performs the command:

`su [-] user -c executable_to_clean_resource`

Note that the agent uses the `su -` option only when the attribute UseSUDash is enabled (1). The UseSUDash attribute is disabled (0) by default.

Action

The various functions of the action entry point are as follows:

- program.vfd
  Checks the availability of the specified program and the execution permissions for the specified program.
- user.vfd
  Checks the existence of the specified user on the host.
- cksum.vfd
  Checks the existence of the same binary on all nodes.
- propcv
  [For internal use only] Invokes the AMF call with arguments to decide whether to allow or prevent processes from starting for an application resource, outside the VCS control, in the cluster. The StartProgram and the processes configured under MonitorProcesses, registered with AMF for offline monitoring, are prevented from starting on the offline node. This helps prevent concurrency violation at an early stage.
- getcksum
  Returns the checksum of the specified program

State definitions for Application agent

ONLINE

Indicates that all processes that are specified in the PidFiles and the MonitorProcesses attribute are running and that the MonitorProgram returns ONLINE.

OFFLINE

Indicates that at least one process that is specified in the PidFiles attribute or MonitorProcesses is not running, or that the MonitorProgram returns OFFLINE.

UNKNOWN	Indicates an indeterminable application state or invalid configuration or that the required attributes have not been configured.
FAULTED	Indicates that the process has terminated unexpectedly or MonitorProgram returns OFFLINE unexpectedly.

Attributes for Application agent on Linux

Table 5-4 Required attributes

Required attribute	Description

StartProgram	The complete path of the executable file that starts the application. The file may be present on local storage or shared storage. Any applicable command-line arguments follow the file path and are separated by spaces.
	For example, the attribute for StartProgram is:
	`/usr/sbin/vxnotify -g dg00 -m >> /var/log/vxnotify.log`
	(and vxnotify is blocking command) set it like:
	`/usr/sbin/vxnotify -g dg00 -m >> /var/log/vxnotify.log &`
	Note: The agent logs the return value of the StartProgram executable. The agent does not treat a non-zero return value as failure of execution and brings the resource online.
	Note: Do not use the opening and closing ({ }) brace symbols in this string.
	Note: In the script, specify a return value that is between 0 and 255.
	Type and dimension: string-scalar
	Example: `"/usr/sbin/sample_app start"`
	In case of RHEL 7, SLES 12, and later distributions, you can choose to start the application processes under system.slice instead of user.slice. To do so, use the `systemctl` command in this attribute value to start the service:
	`"systemctl start appServiceFile"`
	For example:
	`"systemctl start vcs-app1"`

Table 5-4 Required attributes *(continued)*

Required attribute	Description
StartOnly	Indicates whether the application must be monitored or not. If this attribute is set, the agent does not execute the script specified in MonitorProgram, but performs the following actions instead: ■ If the online function is executed and the return code of StartProgram is 0, it reports the ONLINE state. Otherwise, it reports the OFFLINE state. ■ If the offline function is executed and the return code of StopProgram is 0, it reports the OFFLINE state. Otherwise, it reports the ONLINE state. **Note:** If this attribute is set: You must set Critical to 0 so that VCS does not attempt to fail over or take any action if the application faults. You may increase the values of MonitorInterval and OfflineMonitorInterval, because those attributes do not have an impact. Type and dimension: boolean-scalar Default: 0
StopProgram	The complete path of the executable that stops the application. The file may be present on local storage or shared storage. Any applicable command-line arguments follow the file path and are separated by spaces. **Note:** The agent logs the return value of the StopProgram executable. The agent does not treat a non-zero return value as failure of execution and takes the resource offline. **Note:** Do not use the opening and closing ({ }) brace symbols in this string. **Note:** In the script, specify a return value that is between 0 and 255. Type and dimension: string-scalar Example: `"/usr/sbin/sample_app stop"`
At least one of the following attributes: ■ MonitorProcesses ■ MonitorProgram ■ PidFiles	See Table 5-5 on page 225.

Table 5-5 Optional attributes

Optional attribute	Description
CleanProgram	The executable which forcibly stops the application. The Attribute specifies the complete path of the executable. Applicable command line arguments follow the name of the executable. The executable path and argument have spaces separating them. This executable can be on a local storage or on a shared storage. **Note:** Veritas recommends to have the CleanProgram on the local storage so that in case of loss of storage connectivity VCS can take appropriate action to stop the application. **Note:** If the CleanProgram executable returns a non-zero value, the agent treats it as a clean failure and the resource will not fault. Type and dimension: string-scalar Example: `"/usr/sbin/sample_app stop"`
MonitorProcesses	A list of processes that you want monitored and cleaned. Each process name is the name of an executable. Qualify the executable name with its complete path if the path starts the executable. The process name must be the full command line argument that the `ps -u user -eo pid,args` command displays for the process. Type and dimension: string-vector Example: "nmbd"

Table 5-5 Optional attributes *(continued)*

Optional attribute	Description
MonitorProgram	The complete path of the executable file that monitors the application. The file may be present on local storage or shared storage. Any applicable command-line arguments follow the file path and are separated by spaces. MonitorProgram can return the following VCSAgResState values: OFFLINE value is 100 or 1; ONLINE values range from 101 to 110 or 0 (depending on the confidence level); 110 equals confidence level of 100%. Any other value = UNKNOWN. **Note:** Do not use the opening and closing ({ }) brace symbols in this string. If MonitorProgram is configured and not available, then resource state will be: ■ OFFLINE if the resource was in OFFLINE state and not waiting for any action. ■ UNKNOWN if the resource was in any other state or waiting for some action. Type and dimension: string-scalar Example: "/usr/sbin/sample_app_monitor all"
PidFiles	A list of PID (process ID) files that contain the PID of the processes that you want monitored and cleaned. These are application generated files. Each PID file contains one monitored PID. Specify the complete path of each PID file in the list. The process ID can change when the process restarts. If the application takes time to update the PID file, the agent's Monitor function may return an incorrect result. If incorrect results occur, increase the ToleranceLimit in the resource definition. Type and dimension: string-vector

Table 5-5 Optional attributes *(continued)*

Optional attribute	Description
User	The user ID for running StartProgram, StopProgram, MonitorProgram, and CleanProgram. The processes that are specified in the MonitorProcesses list must run in the context of the specified user. Monitor checks the processes to make sure they run in this context. Type and dimension: string-scalar **Note:** If the configured user does not exist or if the home directory is not set for a configured user, the resource state will be UNKNOWN. Default: root Example: user1 **Note:** Alternatively, in case of RHEL 7, SLES 12, and later distributions, you can choose to start the application processes in system.slice instead of user.slice. To do so, set the value of this attribute to **root**. To start an application in user.slice, you can continue to specify the appropriate user name instead of **root**.
EnvFile	The environment file that should get sourced before running any of the StartProgram, StopProgram, MonitorProgram or CleanProgram. Type and dimension: string-scalar Default: "" **Note:** Ensure that the EnvFile adheres the default shell syntax of the configured user. Example: /home/username/envfile
UseSUDash	When the value of this attribute is 0, the agent performs an `su user` command before it executes the StartProgram, the StopProgram, the MonitorProgram, or the CleanProgram agent functions. When the value of this attribute is 1, the agent performs an `su - user` command before it executes the StartProgram, the StopProgram, the MonitorProgram or the CleanProgram agent functions. Type and dimension: boolean-scalar Default: 0 Example: 1

Table 5-5 Optional attributes *(continued)*

Optional attribute	Description
ContainerOpts	ContainerOpts is a resource type attribute. For more information, refer to the **Resource type attributes** section of the *Cluster Server Administrator's Guide - Linux*.

Resource type definition for Application agent

```
type Application (
     static keylist SupportedActions = { "program.vfd",
      "user.vfd", "cksum.vfd", getcksum, propcv }
     static str ArgList[] = { User, StartProgram, StopProgram,
      CleanProgram, MonitorProgram, PidFiles, MonitorProcesses,
      EnvFile, UseSUDash, State, IState, StartOnly }
     static int IMF{} = { Mode = 3, MonitorFreq = 1,
      RegisterRetryLimit = 3 }
     static str IMFRegList[] = { MonitorProcesses, User, PidFiles,
      MonitorProgram, StartProgram, LevelTwoMonitorFreq }
     static int LevelTwoMonitorFreq = 1
     static int ContainerOpts{} = { RunInContainer=1, PassCInfo=0 }
     str User = root
     str StartProgram
     str StopProgram
     str CleanProgram
     str MonitorProgram
     str PidFiles[]
     str MonitorProcesses[]
     str EnvFile
     boolean UseSUDash = 0
     boolean StartOnly = 0
)
```

Notes for Application agent

Using Application agent with IMF

Intelligent monitoring is supported for the Application agent only under specific configurations. The complete list of such configurations is provided in the following table:

Table 5-6

MonitorProgram	MonitorProcesses	PidFiles	IMF Monitoring Mode
Not Configured	Not Configured	Not Configured	Not Applicable
Not Configured	Not Configured	Configured	Online, Offline
Not Configured	Configured	Not Configured	Online, Offline
Not Configured	Configured	Configured	Online, Offline
Configured	Not Configured	Not Configured	Offline Only
Configured	Not Configured	Configured	Online, Offline
Configured	Configured	Not Configured	Online, Offline
Configured	Configured	Configured	Online, Offline

Note: When you do not configure MonitorProcesses, IMF monitors only the StartProgram on the offline node. Hence, the MonitorFreq of IMF attribute must be set to 1 so that IMF monitors the resource on the offline node every monitor cycle.

When multiple processes are configured under the MonitorProcesses attribute and only some of them are running, offline registration with IMF fails repeatedly until RegisterRetryLimit is reached. In such a scenario, IMF cannot determine when the resource goes ONLINE and the agent monitors the resource in the traditional way.

Level two monitoring through MonitorProgram

MonitorProgram can be executed as a second level monitor whereas PidFiles/MonitorProcesses are monitored as first level monitor. To enable level two monitoring for the Application agent, the LevelTwoMonitorFreq attribute of Application type has to be set to a value greater than zero. When configured, the MonitorProgram is executed in monitoring cycles at intervals specified in LevelTwoMonitorFreq attribute.

For example, if j is the value of the MonitorFreq key of the IMF attribute and k is the value of the LevelTwoMonitorFreq attribute, and if the resource is in online state, then traditional monitors for PidFiles/MonitorProcesses run in every j-th monitor cycle and MonitorProgram runs in every k-th monitor cycle.

When MonitorProgram runs as a second level monitor by setting the LevelTwoMonitorFreq value, the limitation of Application agent to leverage IMF for monitoring PidFiles/MonitorProcesses when resource is in online state is overcome.

The processes configured in PidFiles/MonitorProcesses are then registered for IMF monitoring.

If the LevelTwoMonitorFreq attribute is set to zero and when MonitorProgram is configured, then none of the processes specified in PidFiles/MonitorProcesses are registered with IMF for monitoring when the resource is online. In this case, MonitorProgram and the checks for PidFiles and MonitorProcesses execute in every monitor cycle.

LevelTwoMonitorFreq is a type-level attribute. The default value for the LevelTwoMonitorFreq attribute is one (1) so by default MonitorProgram runs as a second level monitor in every monitor cycle. Any changes to this attribute at the Application type level changes the behavior for all application resources.

To modify the LevelTwoMonitorFreq value at type level to a non-default value (for example, 3), execute the following command:

```
# hatype –modify Application LevelTwoMonitorFreq 3
```

If you want to change the LevelTwoMonitorFreq value for selected resources, execute the following commands for each resource in the following sequence. Note that the LevelTwoMonitorFreq value used in the command is only an example.

```
# hares –override app_res_name LevelTwoMonitorFreq

# hares –modify app_res_name LevelTwoMonitorFreq 3
```

The preceding commands override the LevelTwoMonitorFreq attribute at resource level and modify the value of the attribute for a particular resource.

Using Application agent with ProPCV

ProPCV functionality prevents the StartProgram and binary-based processes that are configured under MonitorProcesses from executing on the offline node. This action detects concurrency violation at an early stage in the cycle. However, ProPCV does not prevent script-based processes that are configured under MonitorProcesses from executing on the offline node. Considerations for ProPCV to function:

- You must run the StartProgram with the same order of arguments as configured in the StartProgram attribute. If you change the order of arguments, ProPCV does not prevent the execution of StartProgram. This causes delay in detecting concurrency violation.
  For example, a single command can be run in multiple ways:

  ```
  /bin/tar -c -f a.tar

  /bin/tar -f a.tar -c
  ```

So, ProPCV does not function if you run the command in a way that is not configured in the StartProgram attribute.

- You must start the StartProgram by using the commands or the way specified in StartProgram attribute. But if you use another way or command to start the program that is not specified in the attribute, ProPCV does not prevent the startup of the program. This causes delay in detecting concurrency violation.
- If the StartProgram is a script, do not change the interpreter path in the script file after the StartProgram is registered for offline monitoring. Else, ProPCV may not function for the StartProgram.
- You must not append the StartProgram attribute with the special character **&**. For example, '/app/start.sh &'.

Requirement for programs

The programs specified in StartProgram, StopProgram, MonitorProgram, CleanProgram should not continuously write to STDOUT or STDERR. If required, please redirect STDOUT and STDERR to some file.

Requirement for default profile

The default profile of configured user should not have any blocking command such as `bash` or any other command such as `exec` that changes the behavior of the shell. This may lead to unexpected behavior.

Application monitoring inside Docker container

Veritas InfoScale 7.1 release supports application monitoring inside Docker containers. You must use the `hadockersetup` utility to configure or unconfigure the container resource. You can also use this utility to configure or unconfigure the application resource. The utility is available in the `/opt/VRTSvcs/bin` directory.

Using the hadockersetup utility

Use the `hadockersetup` utility to configure or unconfigure the container resource. The utility is provided along with the agents.

The default path of the utility is: `/opt/VRTSvcs/bin/hadockersetup`

Use the `hadockersetup --help` command to display the help for the utility.

The following is the output:

```
--configure|-C       : Configure the Docker instance service group
--genapp|-g          : Configure the application resource along with Docker
                       instance service group
```

```
--unconfigure|-U     : Unconfigure the Docker instance service group
--help|-h            : Print help message
```

Prerequisites

- The container should be created on the host.
- To monitor the application inside the container, the utility establishes a communication channel between the container and the host. For this purpose, the utility needs VRTSperl, VRTSvcs, VRTSvlic, and VRTSvcsag packages of the same OS as of the container and also the same InfoScale Availability version as of host.
- Disable the firewall.

Configuring

The hadockersetup utility performs the following tasks when you use the `--configure` option:

- Populates the list of containers created on the system. Select the container.
- Adds Docker daemon resource to VCS configuration if not already present.
- Sets communication channel between the host and the container by installing the packages listed in the Prerequisites section. When prompted, enter the path of the packages listed in the Prerequisites section.
- Adds VCS user.
- Adds container resource and sets dependency (online local firm) with the Docker daemon group.

Note: Optional: You can also configure application resource through the utility. The utility will not prompt for confirmation of application resource configuration if you provide `genapp` as the option.

See Sample output of the hadockersetup utility.

Unconfiguring

Use the `--unconfigure` option to unconfigure the container resource along with application resource (if configured).

Note: The VCS user created during configuration will not be removed. You must use the `hauser -delete` *`username`* command to remove the user.

Limitations of the hadockersetup utility

- You will be able to configure application resource only through MonitorProgram. You cannot provide PID or process name.
- You will not be able to reconfigure the container. Unconfigure the resource and then configure the resource again using the utility.
- The utility cannot monitor the same application inside Docker and host if the user is the same. The user must be different.
- If wrong OS or wrong InfoScale Availability version packages are provided, the utility partially installs the packages. You must manually uninstall the partially-installed packages before configuring the container using the utility.
- The application resource does not report the resource as faulted if the user does not exist.
- Unconfiguring the container resource does not remove the VCS user created during configuration. You must use the following command to remove the user:

  `hauser -delete` *`username`*
- Establishing a communication channel between the host and container may fail if firewall is enabled.
- The utility does not verify OS RPMs and RPM versions.
- The utility does not check the correctness of StartProgram/StopProgram provided during application resource configuration.

Requirement for systemd support

In case of the RHEL 7, SLES 12, and later distributions, you can start applications in system.slice instead of user.slice. To do so, you must create a unit service file for your application under `/etc/systemd/system/`. The unit service file name must adhere to the system naming standard and must be unique.

Note: You must reload the systemd daemon after you change the unit configuration as follows:

```
systemctl --system daemon-reload
```

A sample unit service file configuration for the Application agent follows:

Unit service file name:

```
/etc/systemd/system/vcs-app1.service
```

Unit service file contents option 1:

```
[Unit]
Description=Veritas Application service file
Before=vcs.service

[Service]
Type=forking
Restart=no
KillMode=none
ExecStart=appStartCommand >/dev/null 2>&1 </dev/null
User=john
```

where, *appStartCommand* is the command to start the application.

Unit service file contents option 2:

```
[Unit]
Description=Veritas Application service file
Before=vcs.service

[Service]
Type=forking
Restart=no
KillMode=none
ExecStart=/Application/AppStartCustomScript.pl >/dev/null 2>&1 /null
User=john
```

where, `AppStartCustomScript.pl` is the script file that contains the command to start the application and its environment variables that are to be exported.

Sample contents of the `AppStartCustomScript.pl` file:

```
#!/usr/bin/perl

use warnings;
use strict;

system('appStartCommand&');
```

where, *appStartCommand* is the command to start the application.

The following table describes how the entries in the unit service file are interpreted.

Entry	Description
`Before=vcs.service`	This unit service is stopped before the `vcs` service.

Entry	Description
`Restart=no`	This unit service is not started again after it fails.
`KillMode=none`	The kill signal is not delivered to the unit service file to stop the process. Instead, the VCS application agent handles this task using StopProgram or CleanProgram.
`ExecStart=`*`appStartCommand`* `>/dev/null 2>&1 </dev/null` or `ExecStart=`*`appStartScript`* `>/dev/null 2>&1 </dev/null`	Either a direct command that is used to start the application or the path of the script file that contains the command.
`User=john`	The OS user who has privileges to start or stop the application. **Note:** The value of the User attribute of the application resource must be set to **root**. See "Attributes for Application agent on Linux" on page 223.
`EnvironmentFile=/tmp/envfile`	The environment variables file for the application.

Verifying that the application process has started

The following command and its sample output indicate that the application process has been started by the application user that was specified in the unit service file.

```
[root@localhost]# ps -ef | grep -i appStartCommand
```

A sample output is:

```
john 20845 1 0 15:29 ? 00:00:00 appStartCommand
```

Verifying the resource is online

When the resource is online, it appears as follows in the systemd context:

```
[root@localhost]# systemd-cgls
└─system.slice
  ├─sample_app.service
  │ └─20845 appStartCommand
  ├─vcs.service
  │ ├─20280 /opt/VRTSvcs/bin/had
  │ ├─20288 /opt/VRTSvcs/bin/hashadow
```

```
| ├─20307 /opt/VRTSvcs/bin/HostMonitor -type HostMonitor -agdir /
| └─20308 /opt/VRTSvcs/bin/Application/ApplicationAgent -type Application
```

Sample configurations for Application agent

The sample configurations for this agent follow:

Configuration 1 for Application agent

In this example, you configure the executable sample_app as StartProgram and StopProgram, with start and stop specified as command line arguments respectively. Configure the agent to monitor two processes: a process that the app.pid specifies and the process sample_app.

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

Configuration 2 for Application agent

In this example, since no user is specified, it uses the root user. The executable sample_app starts and stops the application using start and stop as the command line arguments. The executable sample_app_monitor monitors the application and uses all as its command line argument. The agent also monitors the sample_app1 and sample_app2 processes.

```
Application samba_app2 (
StartProgram = "/usr/sbin/sample_app start"
StopProgram = "/usr/sbin/sample_app stop"
CleanProgram = "/usr/sbin/sample_app force stop"
MonitorProgram = "/usr/local/bin/sample_app_monitor all"
MonitorProcesses = { "sample_app1", "sample_app2" }
)
```

Configuration 3 for Application agent

The following is a sample configuration for monitoring application inside Docker container:

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

```
cluster dockreg (
        UserNames = {
                 "d_rhel72_container1_dockreg@vcs_lzs@ba38dea0
                   -f66e-11e5-a2ee-af94cf65eb82" = 0 }
        SecureClus = 1
        DefaultGuestAccess = 1
        HacliUserLevel = COMMANDROOT
        )

system Sys1 (
        )

system Sys2 (
        )

group DockerDaemonGRP (
        SystemList = { Sys1 = 0, Sys2 = 1 }
        Parallel = 1
        )

        DockerDaemon DockerDaemonRES (
                )


group rhel72_container1_DockerGRP (
        SystemList = { Sys1 = 0, Sys2 = 1 }
        ContainerInfo = { Name = rhel72_container1, Type = Docker,
         Enabled = 1 }
        Administrators = {
                 "d_rhel72_container1_dockreg@vcs_lzs@ba38dea0
                   -f66e-11e5-a2ee-af94cf65eb82" }
        )

        Application rhel72_container1_ApplicationRES (
                StartProgram = "service1 start"
                StopProgram = "service1 stop"
                CleanProgram = "service1 stop"
                MonitorProgram = "service1 status"
                ContainerOpts = { RunInContainer = 1, PassCInfo = 1 }
                RestartLimit = 3
                )
```

```
        DockerContainer rhel72_container1_DockerRES (
                ContainerName = rhel72_container1
                ContainerInitCommand = bash
                ImageName = "rhel7.2"
                RestartLimit = 3
                )

requires group DockerDaemonGRP online local firm
rhel72_container1_ApplicationRES requires rhel72_container1_DockerRES
```

Sample output of the hadockersetup utility

The following is a sample output of the `hadockersetup --configure` option:

```
Containers not configured under VCS on 'Sys1' are:

        1) rhel72_container1
        2) rhel7C1
        3) registry

Specify the container index: [1]
        Selected container is: 'rhel72_container1'

Specify space separated system names : [Sys1 Sys2]
Configuring DockerDaemon resource to monitor docker daemon process

        Resource 'DockerDaemonRES' added successfully

        Onlining resource 'DockerDaemonRES'

Configuring DockerContainer resource to monitor docker
 container 'rhel72_container1'

        Resource 'rhel72_container1_DockerRES' added successfully

Do you want to online the container resource
 'rhel72_container1_DockerRES'? (Y/N): Y

        Onlining resource 'rhel72_container1_DockerRES' .......... DONE

Specify the RPM source path on host: RPM_Source_Path
        Installing packages in container 'rhel72_container1'
```

```
        Installing VRTSperl ... Done

        Installing VRTSvlic ... Done

        Installing VRTSvcs ... Done

        Installing VRTSvcsag ... Done

        Packages installed successfully inside container 'rhel72_container1'

Specify VCS User for establishing communication channel between host
  and container 'rhel72_container1': [d_rhel72_container1_dockreg]

Specify password (minimum five characters) for above mentioned user:

Do you want to configure application resource to monitor
 application inside container? (Y/N): Y

Configuring application resource 'rhel72_container1_ApplicationRES'

        Resource 'rhel72_container1_ApplicationRES' added successfully

Specify StartProgram for resource rhel72_container1_ApplicationRES:
 /genapp/start_program

Specify StopProgram for resource rhel72_container1_ApplicationRES:
 /genapp/stop_program

Specify CleanProgram for resource rhel72_container1_ApplicationRES:
 /genapp/stop_program

Specify MonitorProgram for resource rhel72_container1_ApplicationRES:
 /genapp/monitor_program

Specify User for resource rhel72_container1_ApplicationRES: [root]

Do you want to configure more containers (Y/N): N
Warning: The container(s) listed below are not configured under VCS.
VCS actions, such as Docker daemon resource offline, may impact
 containers not configured under VCS
        rhel7C1
        registry
```

Configuration 4 for Application agent

In case of RHEL 7, SLES 12, and later distributions, you can choose to start the application processes under system.slice instead of user.slice.

In this example:

- The `vcs-app1` unit service file is used to start the application.
- The `app_stop_prog` script is used to stop the application and clean its resources. These scripts are called from Stop and Clean entry points.
- The `app_monitor_prog` script is used to monitor the application.

```
Application app1 (
    Critical = 0
    User = root
    StartProgram = "/bin/systemctl start vcs-app1"
    StopProgram = "/Application/app_stop_prog.pl stop"
    CleanProgram = "/Application/app_stop_prog.pl kill"
    MonitorProgram = "/Application/app_monitor_prog.pl"
)
```

Configuration 5 for Application agent

In this example, you configure StartProgram and StopProgram for the application and set StartOnly to 1. When you want VCS to only start or stop the application, but not monitor it, you do not have to specify MonitorProgram or MonitorProcess.

```
Application app_res (
    Critical = 0
    StartProgram = "/opt/VRTSperl/bin/perl /start.pl"
    StopProgram = "/opt/VRTSperl/bin/perl /stop.pl"
    StartOnly = 1
)
```

Debug log levels for Application agent

The Application agent uses the following debug log levels:

DBG_1, DBG_2, DBG_3, DBG_4, DBG_5

CoordPoint agent

Use the Coordination Point (CoordPoint) agent to monitor the registrations on the different coordination points on each node.

In addition, the CoordPoint agent monitors changes to the Coordinator Disk Group constitution, such as when a disk is accidently deleted from or added to the Coordinator Disk Group or if the VxVM private region of a disk is corrupted.

The agent also performs detailed monitoring on the CoordPoint resource. You can tune the frequency of the detailed monitoring with the LevelTwoMonitorFreq attribute. For example, if you set this attribute to 5, the agent monitors the Coordinator Disk Group constitution in every fifth monitor cycle.

The CoordPoint agent is a monitor-only agent that runs on each node within the client cluster. It can monitor Coordination Point (CP) servers and SCSI-3 disks.

Coordination Point server as a coordination point

When you have configured a CP server as a coordination point, the CoordPoint agent performs the following tasks:

- Confirms that the client cluster can communicate with the CP server coordination point.
- Validates the node registrations in the CP server database using the `cpsadm` command.

SCSI-3 based disk as a coordination point

In case the coordination point is a SCSI-3 based disk, the CoordPoint agent uses the `vxfenadm` command to confirm that the registered keys on the disk are intact. The Monitor agent function contains the monitoring functionality for SCSI-3 disks and CP servers.

If the agent detects an anomaly, the agent reports it to you so you can repair the coordination point. You may have to perform an online coordinator point replacement procedure if the problem is isolated to the keys registered or you can repair the coordination points automatically (if some keys are missing on one or more coordination points) by configuring the `ActionOnCoordPointFault` attribute. For more information on using the attribute, refer to the Attributes section.

Note: The CoordPoint agent that runs on a given client cluster node monitors the keys for coordination points visible to that node alone.

For important information about this agent, refer to:

See “Notes for the CoordPoint agent” on page 244.

Dependencies

No dependencies exist for the CoordPoint resource.

Agent functions

Monitor	Enables the CoordPoint agent to validate the node registrations in the coordination points and confirms that the coordination points are accessible. In addition, enables the agent to monitor disks in the Coordinator Disk Group. Specifically, if a disk is deleted from or added to the disk group or the VxVM private region of a disk is corrupted. CoordPoint resources are persistent, which means that they cannot be brought online or taken offline. They can only monitor the coordination point registrations. The CoordPoint agent also performs I/O fencing reporting activities. See "CoordPoint agent I/O fencing reporting activities" on page 245.

State definitions

ONLINE	Indicates that the CoordPoint resource is working.
UNKNOWN	Indicates the agent cannot determine the coordination points resource's state. This state may be due to an incorrect configuration.
FAULTED	Indicates that CoordPoint resource is reported for one or more of the following conditions: ■ The number of coordination points with missing keys (or registrations) has exceeded the value of the FaultTolerance attribute. ■ The number of unreachable coordination points has exceeded the value of the FaultTolerance attribute. ■ Coordinator disks are deleted from or added to the Coordinator Disk Group. ■ Public character path of a disk and the device path that corresponds to the device number of that disk in the kernel driver do not match.

Attributes

Table 5-7 Required attribute

Required attribute	Description
FaultTolerance	The FaultTolerance attribute determines when the CoordPoint agent declares that the registrations on the coordination points are missing or connectivity between the nodes and the coordination points is lost. If the number of coordination points with missing keys (or registrations) and or the number of unreachable coordination points exceeds the value of the FaultTolerance attribute, then the agent reports FAULTED. Set the value of this attribute depending on your own configuration requirements. For example, if the FaultTolerance value is set to 1, then the CoordPoint agent reports FAULTED if it sees 2 or more number of coordinator points with missing keys (or registrations) and or the number of unreachable coordination points. Change the value of the FaultTolerance attribute either before the CoordPoint agent starts to monitor or while the CoordPoint agent is monitoring. If the attribute is set while the CoordPoint agent is monitoring, then the CoordPoint agent reads the new value in the next monitor cycle. To view the current FaultTolerance value, enter the following command: # hares -display coordpoint -attribute FaultTolerance Type and dimension: integer-scalar Default: "0"

Table 5-8 Optional attribute

Optional attribute	Description
ActionOnCoordPointFault	This attribute determines whether lost registration keys (if any) on any coordination point can be automatically replaced. It also determines whether to take corrective action if the public character path of a coordinator disk does not match with the device path of that coordinator disk in the kernel driver. By default, the attribute is disabled. To enable the attribute set its value to `RefreshRegistrations`. If the refresh procedure fails twice consecutively, it is not attempted again on that node. You can re-enable the refresh procedure on that node by executing the hares command. `# /opt/VRTS/bin/hares -action coordpoint enable_refresh -sys <sys_name>` Type and dimension: string-scalar Default: None To enable the attribute: Set its value to `RefreshRegistrations`

Resource type definition

```
type CoordPoint (
    static keylist SupportedActions = { enable_refresh }
    static int InfoInterval = 300
    static int OfflineMonitorInterval = 60
    static str ArgList[] = { FaultTolerance,
    ActionOnCoordPointFault, RefreshLockHolderSysName }
    static str Operations = None
    int FaultTolerance
    str ActionOnCoordPointFault = None
    str RefreshLockHolderSysName = None
)
```

Notes for the CoordPoint agent

The notes are as follows:

CoordPoint agent I/O fencing reporting activities

The CoordPoint agent also performs the following I/O fencing reporting activities:

- Checks to determine if I/O fencing is running.
  If I/O fencing is not running, then the CoordPoint agent reports failure.
- Checks the mode of fencing operation. I/O fencing can operate in one of the following three modes:
  - SCSI-3 mode: If I/O fencing runs in SCSI-3 mode, then the CoordPoint agent continues to monitor.
  - Customized mode: If I/O fencing runs in Customized Fencing mode, then the CoordPoint agent continues to monitor.
  - Disabled mode: If I/O fencing runs in disabled mode, no action is required. The CoordPoint agent returns success.
  - Majority mode: If I/O fencing runs in majority mode, no action is required. The CoordPoint agent returns success.

AutoStartList attribute

AutoStartList is a service group attribute that needs to be populated with a system list. The VCS engine brings up the specified service group on the nodes in the list.

AutoStartList is not a required attribute for the service group that contains the CoordPoint resource. The CoordPoint resource is a persistent resource and when a service group is configured with this type of resource, it cannot be brought online.

Specifying the AutoStartList with a system list does not change the behavior of the service group.

Detailed monitoring for the Coordpoint resource

The agent fetches disk names and unique identifiers from the kernel driver for I/O fencing. It checks for disks that are no longer part of the Coordinator Disk Group and also checks for any newly added disks to the disk group. It also compares the public character path of the disks with the device path stored in the kernel driver. The agent faults the resource when any of the checks fail.

The ActionOnCoordPointFault attribute set to RefreshRegistrations

The ActionOnCoordPoint attribute has an impact on the resource state. If the refresh procedure completes successfully, then you might see the resource temporarily going into FAULTED state. In the next monitor cycle, the resource comes back to ONLINE state without any external trigger.

Sample configuration

In this example, the coordination point agent type resource is configured with the value of the FaultTolerance attribute set to 0. At this value setting, the CoordPoint agent reports FAULTED, when the agent determines that at least one coordination point has keys (or registrations) missing and or one coordination point is not reachable.

The following is an example service group (vxfen) extracted from a main.cf file:

```
group vxfen (
    SystemList = { sysA = 0, sysB = 1 }
    AutoFailOver = 0
    Parallel = 1
    AutoStartList = { sysA, sysB }
    )
        CoordPoint coordpoint (
            FaultTolerance=0
            LevelTwoMonitorFreq = 5
            )
    // resource dependency tree
    //
    //     group vxfen
    //     {
    //     CoordPoint coordpoint
    //     }
```

Debug log levels

The CoordPoint agent uses the following debug log levels:

DBG_10

KVMGuest agent

The KVMGuest agent monitors a virtual machine (KVM guest) created in KVM environment or Red Hat Enterprise Virtualization (RHEV) environment.

The agent brings virtual machines online, takes them offline, and also migrates virtual machines. The KVMGuest agent uses virsh commands to manage the virtual machine in KVM environment and Representational State Transfer (REST) APIs to manage the virtual machines in RHEV environment.

Disaster Recovery of virtual machines is supported only in case of RHEV environment.

You can use this agent to make a virtual machine highly available and to monitor it.

Cluster Server supports guest virtual machines created on:

- Red Hat Enterprise Linux 6 update 4, update 5
- Red Hat Enterprise Linux 7

Note: On RHEL 7, the KVMGuest agent supports guest virtual machine created in KVM environment only.

- SuSE Enterprise Linux 11 SP2
- SuSE Enterprise Linux 11 SP3
- Red Hat Enterprise Virtualization 3.3
- Red Hat Enterprise Virtualization 3.4

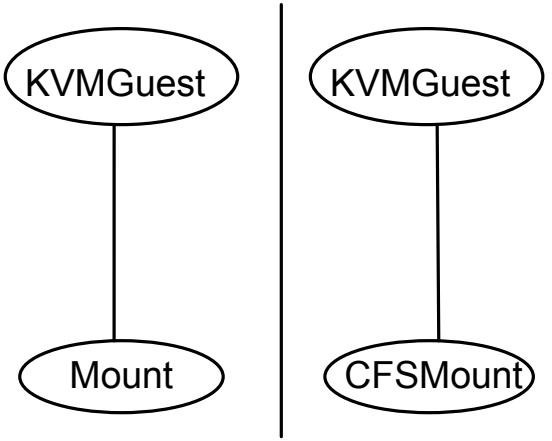
See "Notes for KVMGuest agent" on page 255. for important information on this agent.

Dependencies for KVMGuest agent

KVM Environment: The KVMGuest resource may depend on the Mount resource of CFSMount resource. The dependency is based on where the virtual machine image is located.

RHEV Environment: The KVMGuest resource may not depend on any resource, except in Disaster Recovery (DR) configurations. In a DR configuration, the KVMGuest resource may depend on the appropriate DR replication agent.

Figure 5-3 Sample service group that includes a KVMGuest resource



Agent functions for KVMGuest agent

Online

KVM environment: Agent uses the `virsh start` command to start the guest virtual machine. When the resource is configured to define the guest configuration, agent uses the `virsh define` command to define the virtual machine while bringing it online.

RHEV environment: Agent uses the REST APIs to start the virtual machine. If the DROpts attribute is set to configure the guest network, the agent also sets the payload as a cdrom. This payload contains networking parameters to be set within the guest after a DR failover.

Offline

KVM environment: Agent uses the `virsh shutdown` command to shutdown the guest virtual machine. If the SyncDir attribute is configured to synchronize the guest virtual machine configuration file, then the configuration file is copied to the location configured as a SyncDir attribute.

RHEV environment: Agent uses the REST APIs to shutdown the virtual machine.

Monitor

KVM environment: Agent uses the `virsh domstate` command to determine the status of the guest virtual machine.

RHEV environment: Agent uses the REST APIs to get the status of the virtual machine.

Clean	KVM environment: Agent uses the `virsh destroy` command to forcefully shutdown the guest virtual machine. If the SyncDir attribute is configured to synchronize the guest virtual machine configuration file, then the configuration file is copied to the location configured as a SyncDir attribute. RHEV environment: Agent uses REST APIs to stop the virtual machine.
Open	Starts the vcskvmguestd process which is used to detect the virtual machine migration.
Close	Stops the vcskvmguestd process
Shutdown	Stops the vcskvmguestd process
attr_changed	Checks the value of the DelayAfterGuestOnline and DelayAfterGuestOffline attributes. If it is less than the default value then the warning message is logged. Additionally, it also initiates the virtualization environment validation process whenever any of the following attributes is modified: ■ GuestName ■ RHEVMInfo ■ DRopts
Action	guestmigrated vcskvmguestd process detects the virtual machine migration and executes this action entry point to create the migration state file and initiate the resource monitor using the `hares -probe` command. vmconfigsync: Saves virtual machine configuration of the node to a shared storage. This entry point is only for internal use. Use the havmconfigsync utility for synchronizing virtual machine configuration across the cluster nodes. DevScan: The DevScan action is applicable only in RHEV environments and it is only for internal use. This action is used internally by the preonline trigger script in RHEV DR environments. This action makes sure that Storage Pool Manager (SPM) is active on a healthy node in the current RHEV cluster. After a cluster-wide failover, the state of the replicated devices on RHEL hosts has to be changed from read-only to read-write. This state change is performed by the DevScan action. DevScan also deactivates all the hosts in the remote cluster so that SPM can failover to the local cluster in finite time. DevScan then reactivates all the deactivated hosts. If the DevScan action finds that the current SPM host is in NON_RESPONSIVE state, it isolates that host from RHEV-M so that the SPM status can be moved to some other healthy node in the local cluster.

Migrate	KVM environment: The agent uses the `virsh migrate` command to start virtual machine migration. RHEV environment: The agent uses REST APIs to start virtual machine migration. Additionally, it checks whether the virtual machine migration is allowed or not.

State definitions for KVMGuest agent

ONLINE	Indicates that the virtual machine is running.
OFFLINE	Indicates that the virtual machine has stopped.
FAULTED	Indicates that the virtual machine has failed to start or unexpectedly stopped.
UNKNOWN	Indicates that the problem exists with the configuration or the ability to monitor the resource.
INTENTIONAL OFFLINE	Indicates that the virtual machine is either migrated to another physical host or the guest virtual machine is intentionally suspended by the administrator.

Attributes for KVMGuest agent

Table 5-9 Required attribute

Required attribute	Description
GuestName	The name of the virtual machine created using the KVM hypervisor or RHEV-M. Type and dimension: string-scalar Example: vm1

Table 5-10 Optional attribute

Optional attribute	Description
DelayAfterGuestOnline	Defines the maximum time in seconds that the virtual machine takes to start. You can modify this attribute as required. Type and dimension: integer-scalar Default: 5 Example: 10

Table 5-10 Optional attribute *(continued)*

Optional attribute	Description
DelayAfterGuestOffline	Defines the maximum time in seconds that the virtual machine takes to shut down. You can modify this attribute as required. Type and dimension: integer-scalar Default: 30 Example: 40
SyncDir	Specifies the absolute path of a directory used to synchronize the default configuration directory /etc/libvirt/qemu/ on all the cluster nodes. If this attribute is configured, the online entry point uses the guest configuration file in the specified path to define the guest. This directory must be on the shared storage. This attribute is valid in KVM environment only. Type and dimension: string-scalar Example: /kvmguest_syncdir
GuestConfigFilePath	Specifies the absolute path of the guest configuration file. You can use this attribute as an alternative to SyncDir. If you configure this attribute, ensure that the guest configuration file is available on all the cluster nodes and on same path. VCS uses this path to define the guest on each node. If both SyncDir and GuestConfigFilePath are configured, then SyncDir is preferred over GuestConfigFilePath. This attribute is valid in KVM environment only. Type and dimension: string-scalar Example: /kvmguest/kvmguest1.xml

Table 5-10 Optional attribute *(continued)*

Optional attribute	Description
RHEVMInfo	Specifies information about the RHEV environment The Keys associated with this attribute are: ■ Enabled. Specifies the virtualization environment. If value is: ■ 0: KVM environment ■ 1: RHEV environment ■ Default: 0 ■ URL: Specifies the RHEV-M URL that can be used for REST API communication. Example: https://rhevm-server.domain.com:443 ■ User: Specifies the RHEV-M User that can be used for REST API communication. Examples: admin@internal rhevadmin@example.com ■ Password: Specifies the encrypted password of RHEV-M User. The password should be encrypted using `vcsencrypt` command. ■ Cluster: Specifies the name of the RHEV-M cluster, to which the VCS host belongs. ■ UseManualRHEVMFencing: Enables or disables the use of manual RHEV-M fencing if physical host on which virtual machine is running crashes. The default value is 0. The value 0 signifies that the use of manual RHEV-M fencing is disabled. The value 1 signifies that the use of manual RHEV-M fencing is enabled.
MigrateTimeout	Specifies the timeout value for migrating virtual machines. Default value: 300 seconds
MigrateWaitLimit	Specifies the migrate wait limit. The monitor entry point runs for the number of times specified in the attribute value to determine whether or not the attempted resource migration failed. Default value: 2

Table 5-10 Optional attribute *(continued)*

Optional attribute	Description
DROpts	Defines the DR options. The value of this attribute consists of the following keys: ■ DNSSearchPath: The domain search path used by the virtual machine in this site. The value of this key must contain a list of DNS domain names that are used for the DNS lookup of a hostname in case the domain name of the hostname is not specified. Use spaces to separate the domain names. ■ DNSServers: The list of DNS servers used by the virtual machine in this site. The value of this key must contain a list of IP addresses of DNS servers that are used for the DNS lookup of a hostname. Use spaces to separate the IP addresses. ■ Gateway: The default gateway used by the virtual machine in this site. ■ Device: The Network Interface Card (NIC) that is dedicated to the exclusive IP virtual machine in this site. If this key is not specified, the agent automatically selects the first dedicated NIC for the assignment of IP address, if specified. ■ IPAddress: The IP address to be assigned to the virtual machine in this site after a cross-site failover. ■ Netmask: The netmask to be used by the virtual machine in this site after a cross-site failover. ■ ConfigureNetwork: Setting used by the virtual machine. The DROpts attribute value is applied to the virtual machine only if this key is set to 1. Type and dimension: string-association

Table 5-11 Internal attributes

Internal attribute	Description
IntentionalOffline	For information on how to use the IntentionalOffline resource type attribute, refer to the *Cluster Server Administrator's Guide.*
CEInfo	This is an internal attribute.

Table 5-11 Internal attributes *(continued)*

Internal attribute	Description
ResyncVMCfg	The ResyncVMCfg attribute is set by the havmconfigsync utility. If this attribute is set, the agent redefines the virtual machine configuration if it already exists using the SyncDir attribute. If the SyncDir attribute is not set, GuestConfigFilePath attribute is used. **Note:** You must not set the ResyncVMCfg attribute manually.
AEPTimeout	This is an internal attribute. Do not modify this attribute. This attribute is used to pass the timeout value of agent entry points.

Resource type definition for KVMGuest agent

The resource type definition for the KVMGuest agent is as follows:

```
type KVMGuest (
        static int IntentionalOffline = 1
        static boolean AEPTimeout = 1
        static int MigrateTimeout = 300
        static int MigrateWaitLimit = 2
        static keylist SupportedActions = { "guestmigrated",
        "vmconfigsync", "DevScan" }
        static keylist SupportedOperations = { "migrate"}
        static keylist RegList = { "GuestName", "DelayAfterGuestOnline",
        "DelayAfterGuestOffline", "RHEVMInfo", "DROpts" }
        static str ArgList[] = { GuestName, DelayAfterGuestOnline,
        DelayAfterGuestOffline, SyncDir, GuestConfigFilePath, CEInfo,
        RHEVMInfo, ResyncVMCfg, DROpts }
        str CEInfo{} = { Enabled=0, CESystem=NONE, FaultOnHBLoss=1 }
        str RHEVMInfo{} = { Enabled=0, URL=NONE, User=NONE, Password=NONE,
        Cluster=NONE, UseManualRHEVMFencing=0 }
        str GuestName
        int DelayAfterGuestOnline = 5
        int DelayAfterGuestOffline = 30
        str SyncDir
        str GuestConfigFilePath
        boolean ResyncVMCfg = 0
        str DROpts{} = { ConfigureNetwork=0, IPAddress=NONE, Netmask=NONE,
        Gateway=NONE, DNSServers=NONE, DNSSearchPath=NONE, Device=NONE }
```

Notes for KVMGuest agent

The KVMGuest agent has the following notes:

- Support for guests created on RHEL 6, RHEL 7 (KVM environment), and SuSE Enterprise Linux 11 SP2 and SP3
- Storage and network configurations
- Guest live migration
- Managing virtual machines in RHEV environment
- Using VCS to migrate virtual machines
- Configuring the KVMGuest agent for DR in a global cluster setup
- Configuring a non-admin user for RHEV-M that is using AD-based domain
- Virtual machine failover if host crashes
- KVMGuest agent requires `curl` and `xpath` commands in RHEV environment
- RHEV environment: If a node on which the VM is running panics or is forcefully shutdown, VCS is unable to start the VM on another node

Support for guests created on RHEL 6, RHEL 7 (KVM environment), and SuSE Enterprise Linux 11 SP2 and SP3

The KVMGuest agent supports only those guests that can be created on RHEL 6, RHEL 7 (KVM environment), and SuSE Enterprise Linux 11 SP2 and SP3.

Storage and network configurations

If you have specified customized network and storage configurations for any guest, make sure that same configurations are available on all the cluster nodes.

Guest live migration

For migrating the guest from one node to another node, make sure that the guest image is available on both the nodes simultaneously at the same location. Also, KVM requires that the guest network and storage configuration should be identical on source and destination node. For details, refer to the *Redhat Enterprise Linux Virtualization Guide*.

Managing virtual machines in RHEV environment

Refer to the *Veritas InfoScale Virtualization Guide*.

Managing ISO image in SuSE KVM

By default, KVM guest virtual machines created on SuSE requires the ISO image to be always available while starting the virtual machine. After the virtual machine is created, you can modify the virtual machine configuration to remove the reference to the ISO image. Alternatively, you can configure the VCS Mount resource to make the ISO image available on a cluster node.

Using VCS to migrate virtual machines

Use the `hagrp –migrate` command to initiate the migration of virtual machines. For example:

```
#hagrp –migrate service_group_name -to target_system_name
```

To get the value of the PhysicalServer attribute, enter:

```
#hasys –value system_name PhysicalServer
```

To set the value of the PhysicalServer attribute, enter the following command on each cluster node:

```
#haconf –makerw
```

```
#hasys –modify system_name PhysicalServer "`hostname`"
```

If the PhysicalServer attribute value is not configured, the target system name specified in the `hagrp –migrate` command is considered as the destination system name.

Configuring the KVMGuest agent for DR in a global cluster setup

For information about configuring the KVMGuest agent for DR in a global cluster setup, refer to the *Veritas InfoScale Virtualization Guide*.

Configuring a non-admin user for RHEV-M that is using AD-based domain

RHEV-M can be configured to use AD-based domain or internal domain. The default domain that is set while configuring RHEV-M is internal domain. If RHEV-M is configured to use the internal domain, only an admin user can perform operations. However, if RHEV-M is configured to use an AD-based domain, any user in this particular domain can be configured to perform virtual machine management tasks if that user has the required privileges.

If RHEV-M is configured to use AD-based domain, Veritas recommends that you configure a non-admin user in the User key of the RHEVMInfo attribute. If you

configure an admin user to access RHEV-M that uses AD-based domain, REST APIs may take a long time to respond.

Virtual machine failover if host crashes

The KVMGuest agent may use manual RHEV-M fencing to fence out the host which crashed and initiate virtual machine failover to another active host. For more information, refer to the *Veritas InfoScale Virtualization Guide*.

KVMGuest agent requires `curl` and `xpath` commands in RHEV environment

In a RHEV environment, the KVMGuest agent uses REST APIs for virtual machine management in RHEV-M. The `curl` command is used for initiating REST APIs and the `xpath` command is used for parsing the results returned by the REST APIs. If you enable VCS to manage virtual machines in RHEV environment, you must ensure that the `curl` and `xpath` commands are available on all the hosts. Install the curl package for the `curl` command and install the perl-XML-XPath package for the `xpath` command.

RHEV environment: If a node on which the VM is running panics or is forcefully shutdown, VCS is unable to start the VM on another node

In a RHEV environment, if a node on which a virtual machine is running panics or is forcefully shutdown, the state of that virtual machine is not cleared. RHEV-M sets the VM to UNKNOWN state and VCS is unable to start this virtual machine on another node. You must initiate manual fencing in RHEV-M to clear the state.

This is not a VCS limitation because it is related to RHEV-M design. For more information, refer *Red Hat Enterprise Virtualization Technical Guide*.

To initiate manual fencing in RHEV-M and clearing the VM state

1. In the RHEVMinfo attribute, set the UseManualRHEVMFencing key to 1.

   ```
   UseManualRHEVMFencing = 1
   ```

2. Override the resource attribute:

   ```
   hares -override resource_name OnlineRetryLimit
   ```

3. Modify the OnlineRetryLimit attribute value to 2:

   ```
   hares -modify resource_name OnlineRetryLimit 2
   ```

After you clear the state of the VM, VCS starts the VM on another node.

The following is a sample resource configuration of RHEV-based disaster recovery:

```
group rhev_sg (
        SystemList = { rhelh_a1 = 0, rhelh_a2 = 1 }
        TriggerPath ="bin/triggers/RHEVDR"
        PreOnline=1
        OnlineRetryLimit = 1
        )

KVMGuest rhev_fo (
        RHEVMInfo = { Enabled = 1, URL =
         "https://192.168.72.11:443",
        User = "admin@internal",
        Password = flgLglGlgLglG,
        Cluster = RHEV-PRIM-CLUS,
        UseManualRHEVMFencing = 1 }
        GuestName = swvm02
        OnlineRetryLimit = 2
        )


// resource dependency tree
//
//      group rhev_sg
//      {
//      KVMGuest rhev_fo
//      }
```

Sample configurations for KVMGuest environment

Following are few sample configurations for managing virtual machines in KVM environment.

Sample Configuration 1

In this example, the native LVM volumes are used to store the guest image.

```
group kvmtest1 (
SystemList = { north = 0, south = 1 }
)
KVMGuest res1 (
GuestName = kvmguest1
GuestConfigFilePath = "/kvmguest/kvmguest1.xml"
DelayAfterGuestOnline = 10
DelayAfterGuestOffline = 35
)
```

```
Mount mnt1 (
BlockDevice = "/dev/mapper/kvmvg-kvmvol"
MountPoint = "/kvmguest"
FSType = ext3
FsckOpt = "-y"
MountOpt = "rw"
)
LVMLogicalVolume lv1 (
VolumeGroup = kvmvg
LogicalVolume = kvmvol
)
LVMVolumeGroup vg1 (
VolumeGroup = kvmvg
)
res1 requires mnt1
mnt1 requires lv1
lv1 requires vg1
```

Sample Configuration 2

In this example, the native VxVM volumes are used to store the guest image.

```
group kvmtest2 (
SystemList = { north = 0, south = 1 }
)
KVMGuest res1 (
GuestName = kvmguest1
GuestConfigFilePath = "/kvmguest/kvmguest1.xml"
DelayAfterGuestOnline = 10
DelayAfterGuestOffline = 35
)
Mount mnt1 (
BlockDevice = "/dev/vx/dsk/kvmvg/kvmvol"
MountPoint = "/kvmguest"
FSType = vxfs
FsckOpt = "-y"
MountOpt = "rw"
)
Volume vol1 (
Volume = kvm_vol
DiskGroup = kvm_dg
)
DiskGroup dg1 (
DiskGroup = kvm_dg
```

```
)
res1 requires mnt1
mnt1 requires vol1
vol1 requires dg1
```

Sample Configuration 3

In this example, the native CVM-CFS is used to store the guest image.

```
group cvm (
SystemList = { kvmpm1 = 0, kvmpm2 = 1 }
AutoFailOver = 0
Parallel = 1
AutoStartList = { kvmpm1, kvmpm2 }
)

CFSMount cfsmount (
MountPoint = "/cfsmount"
BlockDevice = "/dev/vx/dsk/cfsdg/cfsvol"
)

CFSfsckd vxfsckd (
)

CVMCluster cvm_clus (
CVMClustName = kvmcfs
CVMNodeId = { kvmpm1 = 0, kvmpm2 = 1 }
CVMTransport = gab
CVMTimeout = 200
)

CVMVolDg cfsdg (
CVMDiskGroup = cfsdg
CVMVolume = { cfsvol }
CVMActivation = sw
)

CVMVxconfigd cvm_vxconfigd (
Critical = 0
CVMVxconfigdArgs = { syslog }
)

cfsmount requires cfsdg
cfsmount requires cvm_clus
```

```
cvm_clus requires cvm_vxconfigd
vxfsckd requires cvm_clus
```

Sample configurations for RHEV environment

Following are few sample configurations for managing virtual machines in RHEV environment.

Sample Configuration 1

A sample configuration with AD-based domain for the KVMGuest agent is as follows:

```
group rhev_grp1 (
SystemList = { north = 0, south = 1 }
)
KVMGuest kvmres1 (
RHEVMInfo = { Enabled = 1,
URL = "https://rhevm-server.example.com:443",
User = rhevadmin@example.com,
Password = bncNfnOnkNphChdHe,
Cluster = dc2_cluster1,
UseManualRHEVMFencing = 0 }
GuestName = rhevvm1
DelayAfterGuestOnline = 20
DelayAfterGuestOffline = 35
)
```

Sample Configuration 2

A sample configuration with internal domain for multiple KVMGuest resources is as follows:

```
group rhev_grp1 (
SystemList = { north = 0, south = 1 }
)
KVMGuest kvmres1 (
RHEVMInfo = { Enabled = 1,
URL = "https://rhevm-server.domain.com:443",
User = admin@internal,
Password = bncNfnOnkNphChdHe,
Cluster = dc2_cluster1,
UseManualRHEVMFencing = 0 }
```

```
GuestName = rhevvm1
DelayAfterGuestOnline = 20
DelayAfterGuestOffline = 35
)

group rhev_grp2 (
SystemList = { north = 0, south = 1 }
)
KVMGuest kvmres2 (
RHEVMInfo = { Enabled = 1,
URL = "https://rhevm-server.domain.com:443",
User = admin@internal,
Password = bncNfnOnkNphChdHe,
Cluster = dc2_cluster1,
UseManualRHEVMFencing = 0 }
GuestName = rhevvm2
DelayAfterGuestOnline = 20
DelayAfterGuestOffline = 35
)
```

Sample Configuration 3

A sample configuration for a KVMGuest resource configured for DR with hardware replication of storage domains is as follows:

```
group Replication (
        SystemList = { node1 = 0 }
        ClusterList = { East = 0, West = 1 }
        Authority = 1
        )

        HTC rhevdr_htc (
                GroupName = rhevdr
                Instance = 1
                )

        // resource dependency tree
        //
        //      group Replication
        //      {
        //      HTC rhevdr_htc
        //      }

group RHEVDR (
```

```
        SystemList = { node1 = 0 }
        )

        KVMGuest hadrev17 (
                RHEVMInfo = { Enabled = 1,
                URL = "https://rhevm-server.domain.com:443",
                User = "admin@internal", Password = iwoUlwL,
                Cluster = RHEV_Prod_Clus,
                UseManualRHEVMFencing = 1 }
                GuestName = hadrev17
                DelayAfterGuestOffline = 100
                DROpts = { ConfigureNetwork = 1,
                IPAddress = "10.209.68.255",
                Netmask = "255.255.255.0",
                Gateway = "10.209.68.1", DNSServers = NONE,
                DNSSearchPath = NONE, Device = eth0 }
                )

        requires group Replication online local firm

        // resource dependency tree
        //
        //      group RHEVDR
        //      {
        //      KVMGuest hadrev17
        //      }
```

Sample Configuration for SuSE KVM

```
group kvmgrp (
      SystemList = { north = 0, south = 1 }
      )
      KVMGuest kvmres1 (
         GuestName = kvmguest1
         DelayAfterGuestOnline = 10
         DelayAfterGuestOffline = 30
         )

requires group mntgrp online local firm

group mntgrp (
      SystemList = { north = 0, south = 1 }
```

```
AutoFailOver = 0
Parallel = 1
AutoStartList = { north, south }
)

Mount mntres1 (
     MountPoint = "/os_iso_image"
     BlockDevice = "nfsserver:/os/suse"
     FSType = nfs
     )
```

Debug log levels for KVMGuest agent

The KVMGuest agent uses the following debug log levels:

DBG_1, DBG_2, DBG_3, DBG_4, DBG_5

Process agent

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

The agent does an exact match of configured PathName and Arguments with the processes in the process table. To clarify, the processes that `ps` command lists should have exact match of PathName and Arguments for all the configured processes in the VCS configuration file `/etc/VRTSvcs/conf/config/main.cf`.

Note that the AMF kernel driver does not monitor kernel processes. Even if you have enabled intelligent monitoring for Process agent, you must use the traditional poll-based monitoring to monitor kernel processes.

IMF awareness

The Process agent is IMF-aware and uses Asynchronous Monitoring Framework (AMF) kernel driver for IMF notification.

For more information about IMF and intelligent resource monitoring, refer to the *Cluster Server Administrator's Guide*.

For more information about IMF-related Process agent functions, see Agent functions for Process agent.

High availability fire drill for Process agent

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

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

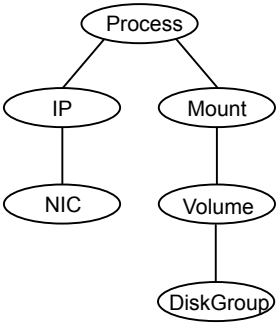
- The existence of a binary executable for the specified process (program.vfd)
- The existence of the same binary on all nodes (program.vfd)

For more information refer to the *Cluster Server Administrator's Guide*.

Dependencies for Process agent

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

Figure 5-4 Sample service group for a Process resource



Agent functions for Process agent

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 terminate, 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.
imf_init	Initializes the agent to interface with the asynchronous monitoring framework (AMF) kernel driver. This function runs when the agent starts up.
imf_getnotification	Gets notification about resource state changes. This function runs after the agent initializes with the AMF kernel driver. The agent continuously waits for notification and takes action on the resource upon notification.
imf_register	Registers the resource entities, which the agent must monitor, with the AMF kernel driver. For example, the function registers the PID for online monitoring of a process. This function runs for each resource after the resource goes into steady state (online or offline).

State definitions for Process agent

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

Attributes for Process agent

Table 5-12 Required attribute for Linux

Required attribute	Description
PathName	Absolute path to access an executable program. This path includes the program name. If a script controls the process, the PathName defines the complete path to the shell. This attribute must not exceed 256 characters. Type and dimension: string-scalar Example: "/usr/sbin/proc1"

Table 5-13 Optional attributes for Linux

Optional attribute	Description
Arguments	Passes arguments to the process. If a script controls the process, the script is passed as an argument. Separate multiple arguments with a single space. A string cannot accommodate more than one space between arguments, nor allow for leading or trailing whitespace characters. Type and dimension: string-scalar
PidFile	The file that contains the process ID for the monitoring process. Specify the PidFile attribute for the monitoring process to use the PID. Otherwise, to complete the monitoring process the agent uses the ps output. Note that when you use scripts, or other indirect mechanisms, to start processes, you must set the PidFile attribute if the ps output is different from the configured values for the PathName or Arguments attributes. Type and dimension: string-scalar Example: "/var/lock/sendmail.pid"
Priority	Priority that the process runs. Priority values range between -20 (highest) to +19 (lowest). Type and dimension: string-scalar Default: 10

Table 5-13 Optional attributes for Linux *(continued)*

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

Note: For a process resource, the combination of PathName and Arguments attributes configured must be exactly the same as that is displayed in the output of the `ps` command.

Resource type definition for Process agent

```
type Process (
        static keylist SupportedActions = { "program.vfd", getcksum }
        static str ArgList[] = { PathName, Arguments, UserName,
        Priority, PidFile }
        static int IMF{} = { Mode = 3, MonitorFreq = 5,
        RegisterRetryLimit = 3 }
        str PathName
        str Arguments
        str UserName = root
        str Priority = 10
        str PidFile
)
```

Usage notes for Process agent

The Process agent has the following notes:

- Prerequisites for processes

Prerequisites for processes

- The processes specified in the PathName attribute must not continuously write to STDOUT or STDERR. If required, redirect STDOUT and STDERR to some file.
- The process must not modify its arguments. If the process modifies its arguments, the Process agent will not be able to monitor the process.

Sample configurations for Process agent

Configuration for Process agent

Configuration for Linux follows:

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

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

Debug log levels for Process agent

The Process agent uses the following debug log levels:

DBG_1, DBG_2, DBG_3, DBG_4, DBG_5

ProcessOnOnly agent

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

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

Dependencies

No child dependencies exist for this resource.

Agent functions

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

State definitions

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

Attributes

Table 5-14 Required attributes for Linux

Required attribute	Description
PathName	Defines absolute path to access an executable program. This path includes the program name. If a process is controlled by a script, the PathName defines the complete path to the shell. The PathName attribute must not exceed 256 characters. The value configured for this attribute needs to match the process listing from the ps output for the agent to display as ONLINE. Type and dimension: string-scalar

Table 5-15 Optional attributes for Linux

Optional attribute	Description
Arguments	Passes arguments to the process. If a process is controlled by a script, the script is passed as an argument. Multiple arguments must be separated by a single space. A string cannot accommodate more than one space between arguments, nor allow for leading or trailing whitespace characters. Type and dimension: string-scalar Example: "-bd -q30m"

Table 5-15 Optional attributes for Linux *(continued)*

Optional attribute	Description
IgnoreArgs	A flag that indicates whether monitor ignores the argument list. ■ If the value is 0, it checks the process pathname and argument list. ■ If the value is 1, it only checks for the executable pathname and ignores the rest of the argument list. Type and dimension: boolean-scalar Default: 0
PidFile	The file that contains the process ID for the monitoring process. Specify the PidFile attribute for the monitoring process to use the Pid. Otherwise, to complete the monitoring process the agent uses the ps output. Note that when you use scripts, or other indirect mechanisms, to start processes, you must set the PidFile attribute when the ps output is different from the configured values for the PathName or Arguments attributes. Type and dimension: string-scalar Example: "/var/lock/sendmail.pid"
Priority	Priority with which the process will run. Priority values range between -20 (highest) to +19 (lowest). Type and dimension: string-scalar Default: 10
UserName	Owner of the process. The process runs with the user ID. Type and dimension: string-scalar Default: root

Resource type definition

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

```
    boolean IgnoreArgs = 0
)
```

ProcessOnOnly agent usage notes

The ProcessOnOnly agent has the following notes:

- Requirement for programs

Requirement for programs

The programs specified in PathName should not continuously write to STDOUT or STDERR. If required, please redirect STDOUT and STDERR to some other file.

Sample configurations

```
group testgrp (
SystemList = { sysA = 0, sysB = 1 }
Parallel = 1
AutoStartList = { sysA, sysB }
OnlineRetryLimit = 3
OnlineRetryInterval = 120
)
Phantom phantom_test (
)
ProcessOnOnly testres (
IgnoreArgs = 1
PathName = "/testApp/testproc"
)
```

Debug log levels

The ProcessOnOnly agent uses the following debug log levels:

DBG_1, DBG_4

AzureAuth agent

To perform any operation on Azure resources, such as updating a resource record set, attaching an Azure data disk, assigning a private IP to a Network Interface, and so on requires you to authenticate that you are an authorized Azure user.

AzureAuth agent authenticates the Azure subscription using service principal credentials.

AzureAuth agent is a persistent resource that monitors the validity of service principal credentials.

Prerequisites

- Create service principal from Azure portal.
  To Create service principal and assign application to role, refer to Azure documentation.
- Ensure that the credentials that are passed on to the AzureAuth agent have at least the minimum required role assigned to service principal.
  The minimum roles required for each agent are:
  - AzureIP: Network Contributor and Virtual Machine Contributor
  - AzureDisk
    - Un-Managed Disks: Virtual Machine Contributor
    - Managed Disks: Contributor
  - AzureDNSZone: DNS Zone Contributor
- Obtain the authentication keys (SubscriptionId, ClientId, SecretKey, and TenantId). See “Obtaining the authentication keys” on page 275.
- Install Python SDK for Azure on all cluster nodes.
  Python SDK can be installed with `pip` as follows:

```
# /opt/VRTSpython/bin/pip3.5 install azure.mgmt.compute==3.0.0
# /opt/VRTSpython/bin/pip3.5 install azure-mgmt-network==1.7.1
# /opt/VRTSpython/bin/pip3.5 install azure.mgmt.dns==1.0.0
# /opt/VRTSpython/bin/pip3.5 install azure.storage==0.33.0
# /opt/VRTSpython/bin/pip3.5 install --force-reinstall
msrest==0.4.29
# /opt/VRTSpython/bin/pip3.5 install --force-reinstall
msrestazure==0.4.19
# /opt/VRTSpython/bin/pip3.5 install --force-reinstall
azure-common==1.1.4
```

  To install Azure Python SDK, the following packages are required:
  - libffi-devel
  - gcc
  - openssl-devel

Dependencies

The AzureAuth agent is not dependent on any other resources.

Agent functions

Monitor	Validates the service principal credentials with Azure.

State definitions

ONLINE	Indicates that the service principal credentials are valid.
UNKNOWN	Indicates that one of the following is true: ■ The service principal credentials are invalid ■ The service principal credentials were modified or deleted after the resource reported ONLINE ■ The service principal credentials are expired

Attributes

Table 5-16 Required attributes

Attribute	Description
SubscriptionId	Identifier that uniquely identifies your Azure subscription. Type and dimension: string-scalar
ClientId	Identifier of the Azure Active Directory (AAD) Application. Type and dimension: string-scalar
SecretKey	Authentication key generated for the AAD application. You must encrypt this secret key using the `vcsencrypt -agent` command. Type and dimension: string-scalar
TenantId	Identifier of the AAD directory in which you created the application. Type and dimension: string-scalar

Resource type definition

```
type AzureAuth (
    static str ArgList[] = { SubscriptionId, ClientId, SecretKey, TenantId }
```

```
    static str Operations = None
    str SubscriptionId
    str ClientId
    str SecretKey
    str TenantId
)
```

Sample configuration

```
AzureAuth Auth_Res (
SubscriptionId = 2dfgg136-fgh6-40dd-b616-c1e9abdf1d63
ClientId = 123456-d10a-4704-8986-beb86739104d
SecretKey = fntPgnUnhTprQrqrnRonSlhPhrQpiNtrItpRhnGrrNklFngLs
TenantId = 12345-0528-4308-brf03-6667d61dd0e3
)
```

Obtaining the authentication keys

1. Log in to the Azure portal.
2. Perform the following to obtain the authentication keys :

Task	Procedure
To obtain the SubscriptionId	1. In the left navigation pane, click **Subscriptions**. A list of your subscriptions is displayed along with the subscription IDs. 2. Copy and provide this Id as SubscriptionId to the AzureAuth agent.
To obtain the TenantID	1. In the left navigation panel, click **Azure Active Directory**. 2. On the page that opens, click **Properties**. 3. Copy the **Directory ID**. 4. Provided this Id as TenantID to the AzureAuth agent.

Task	Procedure
To obtain the ClientId	**1** In the left navigation panel, click **Azure Active Directory**. **2** On the page that opens, click **App registrations**. **3** Search and select your application (service principal) from the list of applications. **4** Copy the **Application ID**. **5** Provide this Id as the ClientId to the AzureAuth agent.
To obtain the SecretKey	**1** From **App registrations** in **Azure Active Directory**, search and select your application. **2** Under **Settings**, select **Keys**. **3** Provide a description of the key, and the expiry duration for the key. **4** Click **Save**. The key is displayed. **Note:** Ensure that you copy and store the key value. You cannot retrieve the key later. **5** Encrypt the key using: `vcsencrypt -agent < Secret Key>`. **6** Provide the encrypted key to the AzureAuth agent as the SecretKey.

Chapter 6

Infrastructure and support agents

This chapter includes the following topics:

About the infrastructure and support agents

Use the infrastructure and support agents to monitor 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. However, in dual stack mode, the NotifierMngr agent can communicate with the SNMP server and SMTP server only if both the servers have both IPv4 and IPv6 IPs enabled on it.

Refer to the *Admin Guide* for a description of types of events that generate notification. See the `notifier`(1) manual page to configure notification from the command line.

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

Other applications with the name notifier can interfere with the NotifierMngr agent. If notifier is started outside VCS control, VCS can only monitor the notifier process if its started with the absolute path. For example, use:

```
# /opt/VRTSvcs/bin/notifier -s m=system_name &
```

system_name is the IP address or host name of the system.

Dependency

The NotifierMngr resource can depend on the NIC resource.

Agent functions

Online	Starts the notifier process with its required arguments.
Offline	VCS sends a `SIGABORT`. If the process does not exit within one second, VCS sends a `SIGKILL`.
Monitor	Monitors the notifier process.
Clean	Sends `SIGKILL`.

State definitions

ONLINE	Indicates that the Notifier process is running.
OFFLINE	Indicates that the Notifier process is not running.
UNKNOWN	Indicates that the user did not specify the required attribute for the resource.

Attributes

Table 6-1 Required attributes for Linux

Required attribute	Description
SnmpConsoles	Specifies the machine names of the SNMP managers and the severity level of the messages to be delivered. The severity levels of messages are `Information`, `Warning`, `Error`, and `SevereError`. Specifying a given severity level for messages generates delivery of all messages of equal or higher severity. SnmpConsoles is a required attribute if SmtpServer is not specified; otherwise, SnmpConsoles is an optional attribute. Specify both SnmpConsoles and SmtpServer if desired. Type and dimension: string-association Example: "172.29.10.89" = Error, "172.29.10.56" = Information
SmtpServer	Specifies the machine name of the SMTP server. SmtpServer is a required attribute if SnmpConsoles is not specified; otherwise, SmtpServer is an optional attribute. You can specify both SmtpServer and SnmpConsoles if desired. Type and dimension: string-scalar Example: "smtp.example.com"

Table 6-2 Optional attributes for Linux

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
MessagesQueue	Size of the VCS engine's message queue. Minimum value is 30. Type and dimension: integer-scalar Default: 30

Table 6-2 Optional attributes for Linux *(continued)*

Optional attribute	Description
NotifierListeningPort	Any valid, unused TCP/IP port number. Type and dimension: integer-scalar Default: 14144
NotifierSourceIP	If this attribute is populated, all the notifications sent from the notifier (SMTP and SNMP) will be sent from the interface having this IP address. **Note:** Make sure that the Source IP given in this attribute is present in the /etc/hosts file or is DNS-resolvable. Type and dimension: string-scalar Example: "10.209.77.111"
SmtpFromPath	Set to a valid email address, if you want the notifier to use a custom email address in the FROM: field. Type and dimension: string-scalar Example: "usera@example.com"

SmtpRecipients	Specifies the email address where SMTP sends information and the severity level of the messages. The severity levels of messages are Information, Warning, Error, and SevereError. Specifying a given severity level for messages indicates that all messages of equal or higher severity are received.
	Note: SmtpRecipients is a required attribute if you specify SmtpServer.
	Type and dimension: string-association
	Example:
	`"james@example.com" = SevereError,`
	`"admin@example.com" = Warning`

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 SMTP VRFY command, then you need to set the SmtpVrfyOff to `1` in order for the SmtpReturnPath value to take effect. Type and dimension: string-scalar Example: "usera@example.com"

Table 6-2 Optional attributes for Linux *(continued)*

Optional attribute	Description
SmtpServerTimeout	This attribute represents the time in seconds notifier waits for a response from the mail server for the SMTP commands it has sent to the mail server. This value can be increased if you notice that the mail server is taking a longer duration to reply back to the SMTP commands sent by notifier. Type and dimension: integer-scalar Default: 10
SmtpServerVrfyOff	Set this value to 1 if your mail server does not support SMTP VRFY command. If you set this value to `1`, the notifier does not send a SMTP VRFY request to the mail server specified in SmtpServer attribute while sending emails. Type and dimension: boolean-scalar Default: 0
SnmpCommunity	Specifies the community ID for the SNMP manager. Type and dimension: string-scalar Default: public
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: string-scalar Default: 162
MessageExpiryInterval	Time in seconds after which the messages expire. If the VCS engine is unable to send a message to the notifier within the message expiry interval, it deletes the message from the VCS engine's message queue. Minimum value: 3600 Type and dimension: integer-scalar Default: 3600

Resource type definition

```
type NotifierMngr (
static int RestartLimit = 3
```

```
static str ArgList[] = { EngineListeningPort, MessagesQueue,
MessageExpiryInterval, NotifierListeningPort, NotifierSourceIP,
SnmpdTrapPort, SnmpCommunity, SnmpConsoles, SmtpServer,
SmtpServerVrfyOff, SmtpServerTimeout, SmtpReturnPath,
SmtpFromPath, SmtpRecipients }
int EngineListeningPort = 14141
int MessagesQueue = 30
int MessageExpiryInterval = 3600
int NotifierListeningPort = 14144
str NotifierSourceIP
int SnmpdTrapPort = 162
str SnmpCommunity = public
str SnmpConsoles{}
str SmtpServer
boolean SmtpServerVrfyOff = 0
int SmtpServerTimeout = 10
str SmtpReturnPath
str SmtpFromPath
str SmtpRecipients{}
)
```

Sample configuration

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

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

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

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

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

Configuration

Configuration for Linux follows:

```
system north

system south

group  NicGrp (
    SystemList = { north, south}
    AutoStartList = { north }
    Parallel = 1
    )

    Phantom my_phantom (
    )

    NIC    NicGrp_eth0 (
        Device = eth0
)
group Grp1 (
    SystemList = { north, south }
    AutoStartList = { north }
    )

    Proxy nicproxy(
    TargetResName = "NicGrp_eth0"
    )

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

    ntfr requires nicproxy

    // resource dependency tree
    //
    //      group Grp1
    //      {
    //      NotifierMngr ntfr
    //              {
```

```
//               Proxy nicproxy
//               }
//       }
```

Debug log levels

The NotifierMngr agent uses the following debug log levels:

DBG_1

Proxy agent

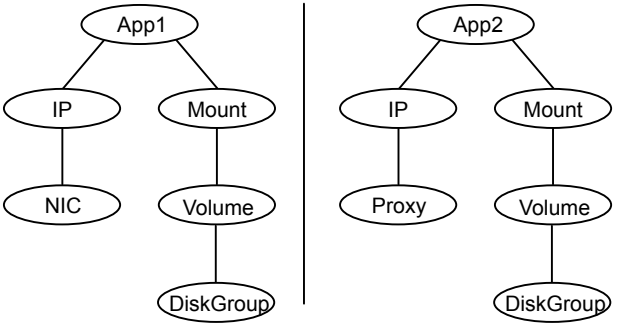
The Proxy agent mirrors the state of another resource on a local or remote system. It provides a means to specify and modify one resource and have its state reflected by its proxies. You can use the agent when you need to replicate the status of a resource.

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

Dependencies

No dependencies exist for the Proxy resource.

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



Agent functions

Monitor	Determines status based on the target resource status.

Attributes

Table 6-3 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: "nic1"

Table 6-4 Optional attribute

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

Resource type definition

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

Sample configurations

Configuration 1

```
Proxy proxy1 (
    TargetResName = "nic1"
)
```

Configuration 2

The proxy resource mirrors the state of the resource nic2 on sysa.

```
Proxy proxy1(
    TargetResName = "nic2"
    TargetSysName = "sysa"
)
```

Configuration 3

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

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

    MultiNICA mnic (
        Device@sysa = { eth0 = "192.123.8.41", eth3 =
        "192.123.8.42" }
        Device@sysb = { eth0 = "192.123.8.43", eth3 =
        "192.123.8.43" }
        NetMask = "255.255.248.0"
    NetworkHosts = { "192.123.10.129", "192.123.10.130" }
    )

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

    ip1 requires mnic


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

        IPMultiNIC ip2 (
```

```
        Address = "192.123.10.178"
        NetMask = "255.255.248.0"
        MultiNICAResName = mnic
        )
    Proxy proxy (
        TargetResName = mnic
        )
ip2 requires proxy
```

Debug log levels

The Proxy agent uses the following debug log levels:

DBG_1, DBG_2

Phantom agent

The agent enables VCS to determine the status of parallel 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 Administrator's Guide* for information on categories of service groups and resources.

Do not use the Phantom resource in failover service groups.

Also, the Phantom resource should not be used in service groups that don't contain any resources.

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

Dependencies

No dependencies exist for the Phantom resource.

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

Agent functions

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

Resource type definition

```
type Phantom (
)
```

Sample configurations

Configuration 1

```
Phantom boo (
)
```

Configuration 2

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

```
include "types.cf"
cluster PhantomCluster
system sysa (
)
system sysb (
)
group phantomgroup (
SystemList = { sysa = 0, sysb = 1 }
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 follow:

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

Veritas supports the RemoteGroup agent when:

- When it points to a global group
  The RemoteGroup agent must then map the state of the global group in the local cluster.
- When it is configured inside a local parallel service group
  The RemoteGroup resources on all cluster nodes monitor the same remote service group unless its attributes are localized.
- When it is configured inside a local failover service group

For more information on the functionality of this agent refer to the *Cluster Server Administrator's Guide*.

Dependency

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

Agent functions

Online	Brings the remote service group online. For more information: See Table 6-5 on page 291.
Offline	Takes the remote service group offline. For more information: See Table 6-5 on page 291.
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. For more information: See Table 6-5 on page 291.
Clean	If the RemoteGroup resource faults, the Clean function takes the remote service group offline. For more information: See Table 6-5 on page 291.

State definitions

ONLINE	Indicates that the remote service group is in an ONLINE state. If the ReturnIntOffline attribute is not set to RemotePartial, then 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. The RemoteGroup resource returns intentional offline if the attribute ReturnIntOffline is set to an appropriate value.
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 6-5 Required attributes

Required attribute	Description
IpAddress	The IP address or DNS name of a node in the remote cluster. The IP address can be either physical or virtual. When configuring a virtual IP address of a remote cluster, do not configure the IP resource as a part of the remote service group. Type and dimension: string-scalar Examples: "www.example.com" or "11.183.12.214"
Port	This is a required attribute when the remote cluster listens on a port other than the default value of 14141. See Table 6-6 on page 294.
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 6-5 Required attributes *(continued)*

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 6-5 Required attributes *(continued)*

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 Veritas 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 Veritas 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 by using the `vcsencrypt` utility. For details, see the *Cluster Server Administrator's Guide*. **Note:** Do not use the vcsencrypt utility when entering passwords from a configuration wizard or the Cluster Manager (Java Console). Type and dimension: string-scalar

Table 6-6 Optional attributes

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

Table 6-6 Optional attributes *(continued)*

Optional attribute	Description
ReturnIntOffline	Select one of the following values for RemoteGroup to return IntentionalOffline: ■ RemotePartial—Indicates that the RemoteGroup resource returns an IntentionalOffline if the remote service group is in an ONLINE|PARTIAL state. ■ RemoteOffline—Indicates that the RemoteGroup resource returns an IntentionalOffline if the remote service group is in an OFFLINE state. ■ RemoteFaulted—Indicates that the RemoteGroup resource returns an IntentionalOffline if the remote service group is OFFLINE|FAULTED. You can use these values in combinations with each other. You must set the IntentionalOffline attribute of the RemoteGroup resource type to 1 for this attribute to work properly. For more information about this attribute, see the *Cluster Server Administrator's Guide*. Type and dimension: string-vector Default: ""
OfflineMonitoringN ode	Defines the cluster node that performs the offline monitoring of the remote service group. This is an internal attribute. Do not modify.

Table 6-7 Type-level attributes

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

Table 6-7 Type-level attributes *(continued)*

Type level attributes	Description
ExternalStateChange	If you want the local service group to go online or offline when the RemoteGroup resource goes online or offline outside VCS control, set the attribute ExternalStateChange appropriately. See the *Cluster Server Administrator's Guide* for more information about these attributes.

Resource type definition

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

Debug log levels

The RemoteGroup agent uses the following debug log levels:

DBG_1

Chapter 7

Testing agents

This chapter includes the following topics:

About the testing agents

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

ElifNone agent

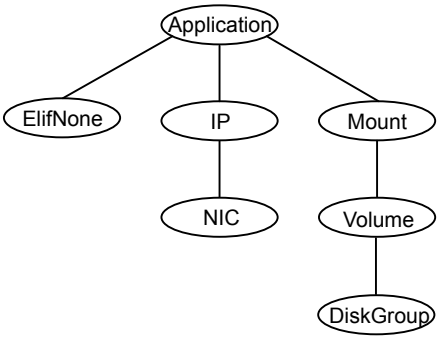
The ElifNone agent monitors a file. It checks for the file's absence.

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

Dependencies for ElifNone agent

No dependencies exist for the ElifNone resource.

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



Agent function for ElifNone agent

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

State definitions for ElifNone agent

ONLINE	Indicates that the file specified in the PathName attribute does not exist.
FAULTED	Indicates that the file specified in the PathName attribute exists.
UNKNOWN	Indicates that the value of the PathName attribute does not contain a file name.

Attributes for ElifNone agent

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 for ElifNone agent

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

Sample configuration for ElifNone agent

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

Debug log levels for ElifNone agent

The ElifNone agent uses the following debug log levels:

DBG_4, DBG_5

FileNone agent

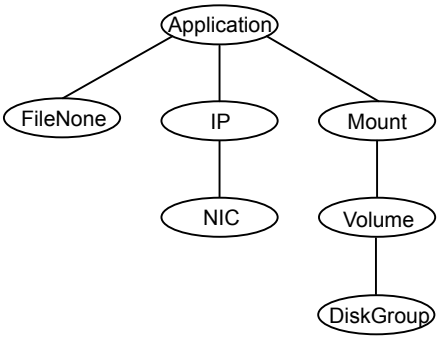
Monitors a file, checks for the file's existence.

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

Dependencies for FileNone agent

No dependencies exist for the FileNone resource.

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



Agent functions for FileNone agent

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

State definitions for FileNone agent

ONLINE — Indicates that the file specified in the PathName attribute exists.

FAULTED — Indicates that the file specified in the PathName attribute does not exist.

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

Attribute for FileNone agent

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 for FileNone agent

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

Sample configuration for FileNone agent

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

Debug log levels for FileNone agent

The FileNone agent uses the following debug log levels:

DBG_4, DBG_5

FileOnOff agent

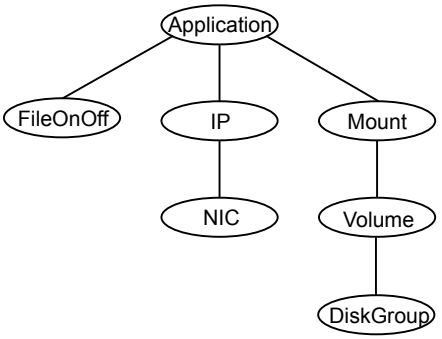
The FileOnOff agent creates, removes, and monitors a file.

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

Dependencies for FileOnOff agent

No dependencies exist for the FileOnOff resource.

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



Agent functions for FileOnOff agent

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	Removes the specified file forcibly when necessary.

State definitions for FileOnOff agent

ONLINE	Indicates that the file specified in the PathName attribute exists.
OFFLINE	Indicates that the file specified in the PathName attribute does not exist.
FAULTED	Indicates that the file specified in the PathName attribute has been removed out of VCS control.
UNKNOWN	Indicates that the value of the PathName attribute does not contain a file name.

Attribute for FileOnOff agent

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 for FileOnOff agent

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

Sample configuration for FileOnOff agent

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

Debug log levels for FileOnOff agent

The FileOnOff agent uses the following debug log levels:

DBG_1, DBG_4, DBG_5

FileOnOnly agent

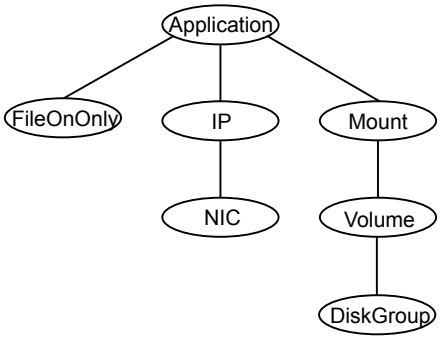
The FileOnOnly agent creates and monitors a file.

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

Dependencies for FileOnOnly agent

No dependencies exist for the FileOnOnly resource.

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



Agent functions for FileOnOnly agent

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

State definitions for FileOnOnly agent

The state definitions for this agent follow:

ONLINE	Indicates that the file specified in the PathName attribute exists.
OFFLINE	Indicates that the file specified in the PathName attribute does not exist and VCS has not attempted to bring the resource online.
FAULTED	Indicates that the file specified in the PathName attribute has been removed out of VCS control.
UNKNOWN	Indicates that the value of the PathName attribute does not contain a file name.

Attribute for FileOnOnly agent

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 for FileOnOnly agent

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

Sample configuration for FileOnOnly agent

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

Debug log levels for FileOnOnly agent

The FileOnOnly agent uses the following debug log levels:

DBG_1, DBG_4, DBG_5

Chapter 8

Replication agents

This chapter includes the following topics:

About the replication agents

Use the replication agents to provide high availability for VVR resources.

Refer to the *Veritas InfoScale Replication Administrator's Guide* for information on configuring the Replication agents for high availability.

RVG agent

Brings the RVG online, monitors read and write access to the RVG, and takes the RVG offline. This is a failover resource. The RVG agent enables replication between clusters. It manages the Primary VVR node in one cluster and the Secondary VVR node in another cluster. Each node can be failed over in its respective cluster. In this way, replication is made highly available.

The RVG agent manages the state of the RVG during local failovers. The RVGPrimary agent manages the role of the RVG during a wide area failover.

Using a VCS global cluster enables you to fail over the Primary role from a Primary VVR node to a Secondary VVR node.

The RVG agent includes the following key features:

- Removes potential single points of failure by enabling Primary and Secondary VVR nodes to be clustered.
- Enables you to bring a service group online to start VCS-managed applications that use VVR.
- Continues replication after a node in a cluster fails without losing updates.
- Ensures that VVR can be added to any VCS cluster by including the RVG resource type definitions.

An example configuration file for this agent that can be used as a guide when creating your configuration is located at:

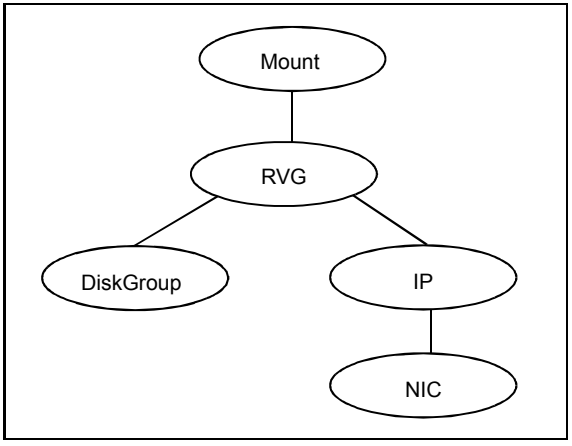
/etc/VRTSvcs/conf/sample_vvr/RVG

Dependencies

The RVG resource represents the RVG (Replicated Volume Group) in the RDS (Replicated Data Set). The RVG resource is dependent on the DiskGroup resource. The RVG resource is also dependent on the IP resources that it uses for replication.

Refer to the *Cluster Server Administrator's Guide* for more information on dependencies.

Figure 8-1 Sample service group for an RVG resource



Agent functions

The RVG agent has the following agent functions:

Online	Verifies whether the DiskGroup agent has recovered the RVG. If not, recovers and starts the data volumes and the Storage Replicator Log (SRL), recovers the RVG, recovers all RLINKs in the RVG, and then starts the RVG.
Offline	Stops the RVG.
Monitor	Monitors the state of the RVG using the `vxprint` command. The RVG resource monitors an RVG for local access only. It does not monitor replication.
Clean	Stops the RVG.
Info	The info entry point displays information about the replication status of a RDS.

State definitions

The RVG agent has the following state definitions:

ONLINE	Indicates that the RVG is in `ENABLED/ACTIVE` state.
OFFLINE	Indicates that the RVG is in `DISABLED/CLEAN` state.
FAULTED	The RVG resource fails if the RVG is not in the ENABLED/ACTIVE state.

Attributes

Table 8-1 Required attributes

Attribute	Description
RVG	The name of the RVG being monitored. Type and dimension: string-scalar Example: "hr_rvg"
DiskGroup	The disk group that this RVG is associated with. Type and dimension: string-scalar Example: "hrbg"

Table 8-1 Required attributes *(continued)*

Attribute	Description
StorageDG	The name of the bunker disk group. Type and dimension: string-scalar Example: "hr_bdg"
StorageRVG	The name of the bunker RVG. Type and dimension: string-scalar Example: "hr_brvg"
StorageHostIds	A space-separated list of the hostids of each node in the bunker cluster. Type and dimension: string-keylist Example: "bunker_host"

Table 8-2 Optional attribute

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

Resource type definitions

The RVG agent resource type definition follows.

```
type RVG (
    static int NumThreads = 1
    static str ArgList[] = { RVG, DiskGroup }
    str RVG
    str DiskGroup
    str StorageRVG
    str StorageDG
    str StorageHostIds
)
```

Sample configurations

```
RVG rvg (
       RVG = ApplicationRVG
       DiskGroup = vvrdg
       StorageRVG = ApplicationRVG
       StorageDG = vvrdg
       StorageHostIds = "bunker_host"
       )
```

RVGPrimary agent

The RVGPrimary agent enables migration and takeover of a VVR Replicated Volume Group (RVG) in a VCS environment. Bringing a resource of type RVGPrimary online causes the RVG on the local host to become a primary.

The agent is useful when hosts in both the primary and secondary side are clustered, in particular a VCS replicated data cluster or a VCS global cluster, to completely automate the availability of writable replicated disks to a VCS-managed application.

The RVGPrimary agent includes the following features:

- Removes the manual steps of migrating a VVR primary and secondary roles when failing over applications across a wide area.
- Minimizes the need for resynchronizing replicated volumes by attempting a migration before attempting a hard takeover.
- Waits for the two sides of a replicated data set to become completely synchronized before migrating roles.
- Supports an automatic fast failback resynchronization of a downed primary if it later returns after a takeover.
- Allows you to distinguish the Primary site after network failure or disaster
- Supports the ability to choose the Primary site after a site failure or network disruption is corrected.
- After a successful migration or takeover of a Secondary RVG, the RVGPrimary agent ensures to automatically start the replication from the new Primary to any additional Secondary(s) that exists in the RDS.
- Before a takeover, the RVGPrimary agent synchronizes the Secondary site with any bunker associated with the Primary site, when the Primary site is not available.

Refer to the *Veritas InfoScale Replication Administrator's Guide* for information on configuring the Replication agents for high availability.

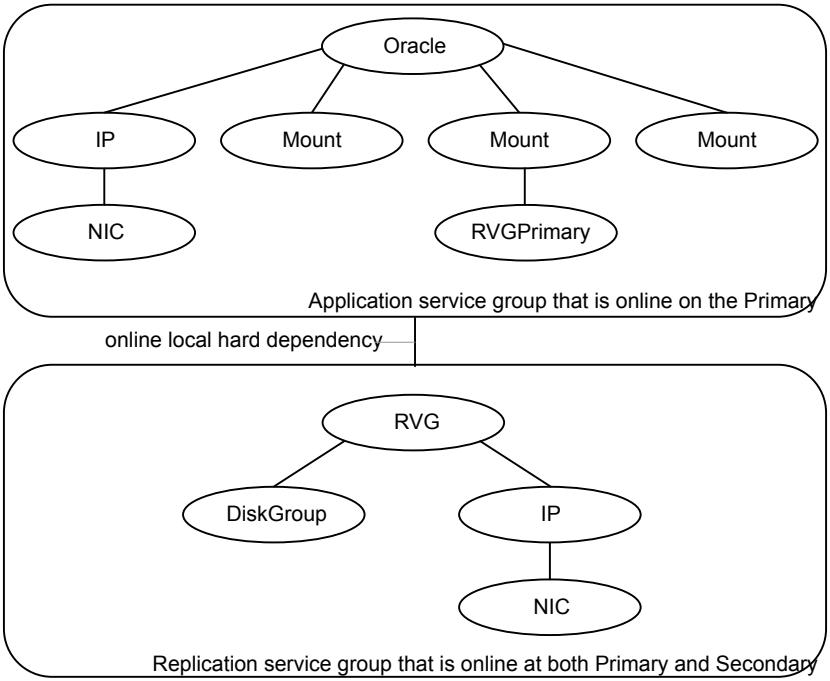
A sample configuration file for this agent that you can use as a guide to create the configuration is located at /etc/VRTSvcs/conf/sample_vvr/RVGPrimary.

Dependencies

You usually use the RVGPrimary agent in conjunction with the RVG agent in two groups with an online local hard group dependency. The parent group contains the resources that manage the actual application and file systems and as the RVGPrimary resource. The child group contains the resources managing the storage infrastructure, which include the RVG and DiskGroup type resources.

Refer to the *Veritas InfoScale Replication Administrator's Guide* for information about the setup of a VVR environment using the RVGPrimary agent.

Figure 8-2 Sample service group for an RVGPrimary resource



Agent functions

The RVGPrimary agent has the following agent functions:

Online	Determines the current role of the RVG. If the role is Secondary it attempts a migration. It waits for any outstanding writes from the original Primary. If the original Primary is down, it attempts a takeover. You can configure the RVGPrimary agent so that, before a takeover, the agent synchronizes the Secondary site with any bunker associated with the Primary site, when the Primary site is not available. If the RVG is a Primary, it performs no actions and goes online.
Offline	Performs no actions.
Monitor	Performs no actions. The RVG agents monitors the actual RVG.
Clean	Performs no actions.
fbsync	This is an action entry point. It resynchronizes the original Primary with the new Primary that has taken over with fast-failback, after the original Primary had become unavailable. This needs to be executed when the original Primary becomes available and starts acting as a Secondary.
ElectPrimary	This is an action entry point. It can be executed to retain the specified RVG as the Primary in a Primary-Primary configuration. For more details, refer to the *Veritas InfoScale Replication Administrator's Guide*.

State definitions

The RVGPrimary agent has the following state definitions:

ONLLINE	Indicates that the role of the RVG is Primary.
FAULTED	The RVG agents monitors the actual RVG. Accidental migration of a VVR Primary outside of VCS causes other resources to fault immediately, such as Mount. No special monitoring by this agent is necessary.

Attributes

Table 8-3 Required attributes

Attribute	Description
RvgResourceName	The name of the RVG resource type that this agent promotes. The name RVG resource type which has been configured using the RVG agent. Type and dimension: string-scalar
AutoTakeover	A flag to indicate whether the agent should perform a takeover on online if the original Primary is down. AutoTakeover and AutoResync are mutually exclusive attributes. When AutoTakeover=0, the primary-elect feature is not applicable; therefore, it is not supported. Type and dimension: integer-scalar

Table 8-3 Required attributes *(continued)*

Attribute	Description
AutoResync	Indicates whether the agent should attempt to automatically perform a fast-failback resynchronization of the original Primary after a takeover and after the original Primary returns. You can use the following values for this attribute: ■ 0–instructs the agent to not attempt to perform a fast-failback resynchronization of the original Primary after a takeover and after the original Primary returns. ■ 1–instructs the agent to attempt to automatically perform a fast-failback resynchronization of the original Primary after a takeover and after the original Primary returns. ■ 2–instructs the agent to use the primary-elect feature. The agent does not attempt to perform a fast-failback resynchronization of the original Primary after a takeover and after the original Primary returns. The RVGPrimary agent also creates space-optimized snapshots for all the data volumes in the RVG resource. If you set the AutoResync attribute to 2 (to enable the primary-elect feature) the value of the BunkerSyncTimeOut attribute must be zero to disable the automated bunker replay feature. You cannot use the automated bunker replay feature and the primary-elect feature in the same environment. AutoTakeover and AutoResync are mutually exclusive attributes. When AutoTakeover=0, the primary-elect feature is not applicable; therefore, it is not supported. Type and dimension: integer-scalar

Table 8-3 Required attributes *(continued)*

Attribute	Description
BunkerSyncTimeOut	The value for the BunkerSyncTimeOut attribute determines if you want the bunker to perform a replay or not. You set the value in seconds for the time that you want to allot for the replay. Use one of the following values for the BunkerSyncTimeOut attribute: ■ If you do not use a value for this attribute (the default null value), the RVGPrimary agent considers it an infinite timeout value. The agent replays all the writes on the Bunker Replicator Log to the Secondary. Only after the agent sends all the writes, VCS performs the takeover on the Secondary. ■ If you set the value for this attribute to 0, you disable bunker replay for the agent. The RVGPrimary agent immediately performs a takeover on the Secondary. The agent does not send pending writes from the Bunker to the Secondary. ■ If you set the value to a number of seconds, then the RVGPrimary agent sends writes for that amount of time to the Secondary. After the agent meets the time limit, it performs the takeover on the Secondary. The bunker replay time in this case is equal to the value in seconds. You can set this value dynamically. The RVGPrimary agent's OnlineTimeout and OnlineRetryLimit attribute values determine the available time for an RVGPrimary resource to complete its online operation. Use the following formula to get the Time Available for Online to Complete (TAOC): TAOC = (OnlineTimeout + (OnlineRetryLimit * OnlineTimeout))

Table 8-3 Required attributes *(continued)*

Attribute	Description
BunkerSyncTimeOut (cont.)	When you set the BunkerSyncTimeOut value in seconds, the value of TAOC for the RVGPrimary agent should be greater than the desired BunkerSyncTimeOut value. Using a TAOC value that is greater than BunkerSyncTimeOut value ensures that the bunker replay and the RVG takeover can complete in the allotted time for that particular online operation. If the TAOC is smaller than BunkerSyncTimeOut value and the bunker replay does not complete within the allotted time for the online process, the resource faults. If the resource faults, clear the fault. Try the online operation again if the resource has not failed over to other cluster node in the configuration. If you increase the value of the BunkerSyncTimeOut attribute, you need to increase the value of the OnlineTimeout or OnlineRetryLimit attribute so that TAOC remain greater than changed value. This is to ensure to have bunker replay completed within allotted time for online. If the value of the AutoResync attribute is 2, you must set the value of the BunkerSyncTimeOut attribute to 0 (to disable automated bunker replay). Type and dimension: string-scalar Default value: ""

Table 8-4 Optional attributes

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

Table 8-4 Optional attributes *(continued)*

Attribute	Description

ResyncType	Allows you to choose between automatic synchronization and difference-based synchronization.
	By default, difference-based synchronization is enabled for resynchronization.
	To enable automatic synchronization, run the following commands:
	`# haconf -makerw`
	`# hares -modify RVGPrimary_resource_name ResyncType 1`
	`# haconf -dump -makero`
	`# hares -value RVGPrimary_resource_name ResyncType`
	To track automatic synchronization progress, run the following commands:
	`# vxrlink -g dg_name -i time_interval status rlk_name`
	`# vradmin -g dg_name repstatus rvg_name`
	Type and dimension: integer-scalar
	Default: 0

Table 8-5 Internal attribute

Attribute	Description
BunkerSyncElapsedTime	For internal use only, do not modify. This value in seconds signifies the amount of time that a Secondary RVG has waited for synchronization from the bunker host to complete. Type and dimension: integer-scalar

Note: Default settings of AutoTakeover=1 and AutoResync=0 cause the first failover to succeed when the original Primary goes down, and upon the original Primary's return, the RDS (Replicated Data Set) has a Primary-Primary configuration error. Set the default value of the AutoResync attribute of the RVGPrimary and RVGSharedPri agents to 1 if you want the agent to automatically attempt a fast-failback resynchronization of the original Primary after a takeover and after the original Primary returns. This prevents the Primary-Primary configuration error.

Resource type definitions

The RVGPrimary resource type definition follows.

```
type RVGPrimary (
    static keylist SupportedActions = { fbsync, electprimary }
    static int NumThreads = 1
    static int OnlineRetryLimit = 1
    static str ArgList[] = { RvgResourceName, "RvgResourceName:RVG",
    "RvgResourceName:DiskGroup", AutoTakeover, AutoResync,
    BunkerSyncTimeOut, BunkerSyncElapsedTime }
    str RvgResourceName
    int AutoTakeover = 1
    int AutoResync = 0
    int ResyncType = 0
    str BunkerSyncTimeOut
    int BunkerSyncElapsedTime = 0
)
```

Sample configurations

```
RVGPrimary rvg-pri (
  RvgResourceName = rvgRes
 )
```

RVGSnapshot

For a fire drill, creates and destroys a transactionally consistent space-optimized snapshot of all volumes in a VVR secondary replicated data set. The RVGSnapshot agent takes space-optimized snapshots on a secondary RVG. These snapshots can be mounted and written to without affecting the actual replicated data, which means that the space-optimized snapshot can be an effective tool for scheduling a “fire drill” to confirm that a wide-area failover is possible. By combining this agent with the VCS Mount agent, the CFSMount agent, and VCS agents that manage the application being replicated, you can create a special fire drill service group. You can bring this service group online and take it offline at regularly scheduled intervals to verify that the disaster recovery environment is robust.

In addition to the agent itself, a text-based wizard /opt/VRTSvcs/bin/fdsetup that prepares the VVR and VCS infrastructure for a fire drill and a script /opt/VRTSvcs/bin/fdsched that runs the fire drill and consolidates the results are also included.

Complete details are in the *Cluster Server Administrator's Guide*.

The RVGSnapshot agent includes the following key features:

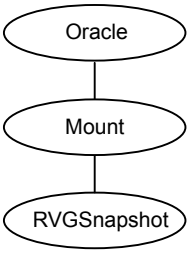
- Automates the process of creating a space-optimized snapshot on a VVR secondary that can be mounted to simulate a wide-area failover without affecting the production application.
- Includes a wizard to effectively set up and schedule fire drills that are completely managed by VCS.

Note: The RVGSnapshot agent does not support Volume Sets.

Dependencies

The RVGSnapshot agent depends on these resources.

Figure 8-3 Sample service group for an RVGSnapshot resource



Agent functions

The RVGSnapshot agent has the following agent functions:

Online	Creates a transactionally consistent snapshot of all volumes in the RVG.
Offline	Destroys the snapshot.
Monitor	No operation; failure of the snapshot will be indicated by the failure of the Mount resource of any file systems mounted on it.
Clean	Cleans up any failed snapshot creation or deletion.

State definitions

The RVGSnapshot agent has the following state definitions:

ONLINE	Indicates that a snapshot was created.
OFFLINE	Indicates that a snapshot was destroyed.

FAULTED The RVGSnapshot resource faults on timeout if a snapshot creation did not succeed during an online.

Attributes

Table 8-6 Required attributes

Attribute	Description
RvgResourceName	The name of the VCS RVG-type resource that manages the RVG that will be snapshot by this agent. Type and dimension: string-scalar
CacheObj	Name of the cache object that is required for a space-optimized snapshot; the fdsetup wizard will create one if it does not exist Type and dimension: string-scalar
Prefix	Token put before the name of the actual volume when creating the snapshotted volumes. Type and dimension: string-scalar

Table 8-7 Optional attributes

Attribute	Description
DestroyOnOffline	A flag to indicate whether to destroy the snapshot upon taking the resources offline. For a fire drill, the snapshot should be deleted to reduce any performance impact of leaving the snapshot for a long period of time; however, if there is interest in keeping the data, then this value should be set to 0. The default is 1 (true). Type and dimension: integer-scalar Default: 1
FDFile	The fire drill schedule updates this attribute with the system name and the path to a file containing the output of the last complete fire drill for the group containing an RVGSnapshot resource. Type and dimension: string-scalar

Table 8-7 Optional attributes *(continued)*

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

Resource type definitions

The resource type definition for the RVGSnapshot agent follows.

```
type RVGSnapshot (
   static keylist RegList = { Prefix }
   static int NumThreads = 1
   static str ArgList[] = { RvgResourceName, CacheObj, Prefix,
   DestroyOnOffline }
   str RvgResourceName
   str CacheObj
   str Prefix
   boolean DestroyOnOffline = 1
   temp str FDFile
   temp str VCSResLock
)
```

Sample configurations

```
RVGSnapshot rvg-sos (
       RvgResourceName = ApplicationRVG
       CacheObj = cacheobj
        Prefix = snap
        )
```

RVGShared agent

Monitors the RVG in a shared environment. This is a parallel resource. The RVGShared agent enables you to configure parallel applications to use an RVG in a cluster. The RVGShared agent monitors the RVG in a shared disk group environment. The RVGShared agent must be configured as a parallel group in VCS. Typically, the RVGShared resource is online or offline at the same time on all the nodes in the VCS cluster. An example configuration file for this agent that can be used as a guide when creating your configuration is located at /etc/VRTSvcs/conf/sample_vvr/RVGLogowner.

Dependencies

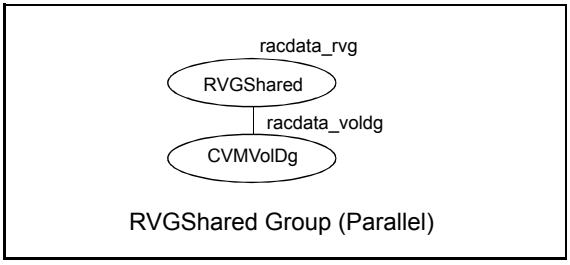
The RVGShared resource represents the RVG of the RDS. The RVGShared resource is dependent on the CVMVolDg resource.

The RVGShared resource must be configured in a parallel group.

Refer to the *Veritas InfoScale Replication Administrator's Guide* for information on configuring parallel applications for highly availability.

Refer to the *Cluster Server Administrator's Guide* for more information on dependencies.

Figure 8-4 Sample service group for an RVGShared resource



Note: Do not add any volumes that are part of the RVG in the CVMVolume attribute of the CVMVolDg resource. The volumes in the RVG are managed by the RVGShared resource.

Agent functions

The RVGShared agent has the following agent functions:

Online	Verifies whether the RVG is started. If the RVG is not started, recovers and starts the RVG.
Offline	No action.
Monitor	Displays the state as `ONLINE` if the RVG is started. Displays the state as `OFFLINE` if the RVG is not started.
Clean	No action.
Info	The info entry point displays information about the replication status of a RDS.

State definitions

The RVGShared agent has the following state definitions:

ONLINE	Indicates that the RVG is in the `ENABLED/ACTIVE` state.
OFFLINE	Indicates that the RVG is not in the `ENABLED/ACTIVE` state or that the administrator has invoked the offline entry point.

Attributes

Table 8-8 Required attributes

Attribute	Description
RVG	The name of the RVG being monitored. Type and dimension: string-scalar
DiskGroup	The shared-disk group with which this RVG is associated. Type and dimension: string-scalar

Table 8-9

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

Resource type definitions

The RVGShared resource type definition follows.

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

Sample configurations

```
RVGShared racdata_rvg (
RVG = rac1_rvg
DiskGroup = oradatadg
)
```

RVGLogowner agent

Assigns and unassigns a node as the logowner in the CVM cluster; this is a failover resource. The RVGLogowner agent assigns or unassigns a node as a logowner in the cluster. To replicate data, VVR requires network connectivity between the Primary and the Secondary. In a shared disk group environment, only one node, that is, the logowner, can replicate data to the Secondary.

For replication to be highly available, the logowner must be highly available. To make the logowner highly available, the RVGLogowner resource must be configured as a resource in a failover group. Also, a virtual IP must be set up on the logowner to enable replication and failover of the logowner from one node to another in a cluster. The virtual IP must be configured as an IP resource.

For more information about the logowner, see the *Veritas InfoScale Replication Administrator's Guide*. An example configuration file for this agent that can be used as a guide when creating your configuration, is located at /etc/VRTSvcs/conf/sample_vvr/RVGLogowner.

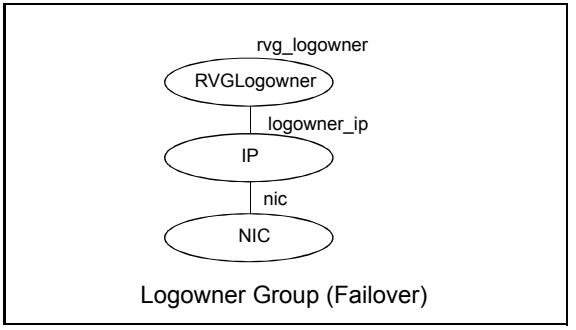
Dependencies

The RVGLogowner resource represents the logowner for RVG in the cluster. The RVGLogowner resource is dependent on the IP resource that it uses for replication.

The RVGLogowner resource must be configured in a failover group. The RVGLogowner group is used in conjunction with the RVGSharedPri and RVGShared agents in separate groups, with the appropriate service group dependencies.

For more information on dependencies, refer to the *Cluster Server Administrator's Guide*

Figure 8-5 Sample service group for an RVGLogowner resource



Agent functions

The RVGLogowner agent has the following agent functions:

Online	Assigns the logowner on the node.
Offline	Unassigns the logowner on the node.
Monitor	Returns `ONLINE` if the node is the logowner and the RVG is in `ENABLED/ACTIVE` state. Returns `OFFLINE` if the node is the logowner and the state is not `ENABLED/ACTIVE`, or if the node is not the logowner (regardless of the state). The RVG for which the logowner is monitored must be configured as the RVGShared resource type.
Clean	Unassigns the logowner on the node.

State definitions

The RVGLogowner agent has the following state definitions:

ONLINE	Indicates that the node is the logowner for the RVG in the cluster.
OFFLINE	Indicates that the node is not the logowner for the RVG in the cluster.

Attributes

Table 8-10 Required attributes

Attribute	Description
RVG	The name of the RVG being monitored. Type and dimension: string-scalar Example: "hr_rvg"
DiskGroup	The disk group with which this RVG is associated. Type and dimension: string-scalar Example: "hrbg"

Table 8-11 Optional attribute

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

Table 8-12 Internal attributes

Attribute	Description
StorageDG	For internal use only, do not modify. The name of the bunker disk group. Type and dimension: string-scalar Example: "hr_bdg"

Table 8-12 Internal attributes *(continued)*

Attribute	Description
StorageRVG	For internal use only, do not modify. The name of the bunker RVG. Type and dimension: string-scalar Example: "hr_brvg"
StorageHostIds	For internal use only, do not modify. A space-separated list of the host IDs of each node in the bunker cluster. Type and dimension: string-keylist Example: "bunker_host"

Resource type definitions

The RVGLogowner resource type definition follows.

```
type RVGLogowner (
    static int NumThreads = 1
    static str ArgList[] = { RVG, DiskGroup }
    static int OnlineRetryLimit = 5
    str RVG
    str DiskGroup
    str StorageRVG
    str StorageDG
    str StorageHostIds
)
```

RVGLogowner agent notes

The RVGLogowner agent has the following notes:

Sample configurations

```
RVGLogowner vvr_rvglogowner (
  RVG = app_rvg
  DiskGroup = vvrdg
  )
```

RVGSharedPri agent

Attempts to migrate or takeover a Secondary to a Primary when a parallel service group fails over. The RVGSharedPri agent enables migration and takeover of a VVR Replicated Data Set (RDS) in parallel groups in a VCS environment. Bringing a resource of type RVGSharedPri online causes the RVG on the local host to become a primary if it is not already. The agent is useful when hosts in both the primary and secondary side are clustered using a VCS global cluster, to completely automate the availability of writable replicated disks to an application managed by VCS.

You cannot use the primary-elect feature with this agent. For a detailed description of the primary-elect feature, see *Veritas InfoScale Replication Administrator's Guide*.

The RVGSharedPri agent includes the following key features:

- Removes manual steps of migrating a VVR primary and secondary roles when failing over applications across a wide area.
- Minimizes the need for resynchronizing replicated volumes by attempting a migration before attempting a hard takeover.
- Waits for the two sides of a replicated data set to become completely synchronized before migrating roles.
- Supports an automatic fast failback resynchronization of a downed primary if it later returns after a takeover.
- After successful migration or takeover of a Secondary RVG, the agent automatically starts the replication from the new Primary to any additional Secondary(s) that exists in the RDS.

Sample configuration files are located in the `/etc/VRTSvcs/conf/sample_rac/` directory and include `CVR` in the filename. These sample files are installed as part of the `VRTSdbac` RPM, and can be used as a guide when creating your configuration.

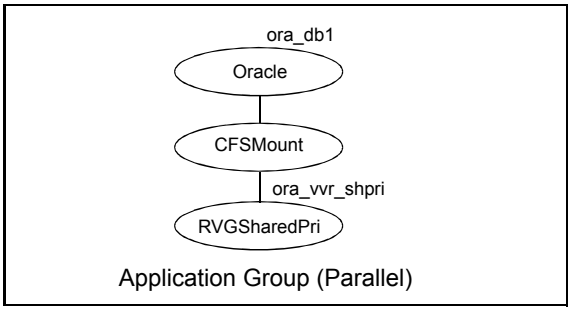
Dependencies

The RVGSharedPri agent is used in conjunction with the RVGShared and RVGLogowner agents in separate groups, with the appropriate service group dependencies.

Refer to the *Veritas InfoScale Replication Administrator's Guide* for information on configuring parallel applications for highly availability.

The RVGSharedPri agent must be configured in a parallel service group. The application service group contains the resources managing the actual application and file systems as well as the RVGSharedPri agent.

Figure 8-6 Sample service group for an RVGSharedPri resource



Agent functions

The RVGSharedPri agent has the following agent functions:

Online	Determines the current role of the RVG; if Secondary, attempt a migrate, waiting for any outstanding writes from the original Primary; if the original Primary is down attempt a takeover; if the RVG is a Primary, perform no actions and go online
Offline	Performs no actions.
Monitor	Performs no actions; monitoring of the actual RVG is done by the RVGShared agent.
Clean	Performs no actions.
fbsync	This is an action entry point. It resynchronizes the original Primary with the new Primary that has taken over with fast-failback, after the original Primary had become unavailable. This needs to be executed when the original Primary becomes available and starts acting as a Secondary.
resync	This is an action entry point. It resynchronizes the Secondaries with the Primary using DCM.

State definitions

The RVGSharedPri agent has the following state definitions:

FAULTED Monitoring of the actual RVG is done by the RVGShared agent; accidental migration of a VVR Primary outside of VCS would cause other resources to fault immediately, such as Mount, so no special monitoring by this agent is necessary.

Attributes

Table 8-13 Required attributes

Attribute	Description
RvgResourceName	The name of the RVGShared resource type that this agent will promote, that is, the name RVG resource type which has been configured using the RVGShared agent. Type and dimension: string-scalar
AutoTakeover	A flag to indicate whether the agent should perform a takeover on online if the original Primary is down. Type and dimension: integer-scalar Default: 1
AutoResync	A flag to indicate whether the agent should attempt to automatically perform a fast-failback resynchronization of the original Primary after a takeover and after the original Primary returns. Type and dimension: integer-scalar Default: 0
VCSResLock	This attribute is reserved for internal use by VCS. Type and dimension: string-scalar

Table 8-14 Optional attribute

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

Table 8-14 Optional attribute *(continued)*

Attribute	Description

ResyncType	Allows you to choose between automatic synchronization and difference-based synchronization.
	By default, difference-based synchronization is enabled for resynchronization.
	To enable automatic synchronization, enter:
	`# haconf -makerw`
	`# hares -modify RVGSharedPri_resource_name ResyncType 1`
	`# haconf -dump -makero`
	`# hares -value RVGSharedPri_resource_name ResyncType`
	To track automatic synchronization progress, enter:
	`# vxrlink -g dg_name -i time_interval status rlk_name`
	`# vradmin -g dg_name repstatus rvg_name`
	Type and dimension: integer-scalar
	Default: 0

Note: Default settings of AutoTakeover=1 and AutoResync=0 cause the first failover to succeed when the original Primary goes down, and upon the original Primary's return, the RDS has a Primary-Primary configuration error. Set the default value of the AutoResync attribute of the RVGPrimary and RVGSharedPri agents to 1 if you want the agent to automatically attempt a fast-failback resynchronization of the original Primary after a takeover and after the original Primary returns. This prevents the Primary-Primary configuration error.

Resource type definitions

The RVGSharedPri resource type definition follows.

```
type RVGSharedPri (
    static keylist SupportedActions = { fbsync, resync }
    static int NumThreads = 1
    static int OnlineRetryLimit = 1
    static str ArgList[] = { RvgResourceName, "RvgResourceName:RVG",
    "RvgResourceName:DiskGroup", AutoTakeover, AutoResync }
    str RvgResourceName
    int AutoTakeover = 1
    int AutoResync = 0
```

```
    int ResyncType = 0
    temp str VCSResLock
)
```

Sample configurations

```
RVGSharedPri ora_vvr_shpri (
RvgResourceName = racdata_rvg
OnlineRetryLimit = 0
)
```

VFRJob agent

Veritas File Replicator Job (VFRJob) agent provides high availability for Veritas File System Replicator Job (VFR Job). VFR Job schedules replication of file systems from a source system to a target system. The agent makes the VFR Job highly available on a source system. VFR Job supports replication of VxFS and CFS type file systems.

Refer to *Veritas InfoScale Replication Administrator's Guide* for more details.

Overview

The VFRJob Agent starts scheduling of VFR Job, monitors VFR Job status, and stops scheduling of VFR Job.

Use the VFRJob agent to make the replicator job highly available on a source system if the source system faults. The VFRJob type resource is a failover resource and provides high availability (HA) for VFR Job. It monitors the VFR Job on the source system. The source system is where the file system is mounted and the file system is replicated from the source system to the target system. The target system, where the file system is replicated to, must be outside of the cluster of the source system. Target system can be part of a different cluster.

If the system performing file system replication (system that hosts the file system) faults, the file system fails over to another system in the cluster. And the VFRJob resource also fails over to that system. Thus VFRJob agent makes the VFR Job highly available.

VFRJob depends on vxfstaskd daemon and vxsfsrepld daemon. The vxfstaskd daemon schedules the replicator job while the vxfsrepld daemon functions as the replicator daemon. Both the daemons should be running on source and target systems.

Refer to the sample configuration that shows how VCS provides high availability for the scheduler daemon and the replication daemon on the source as well as the target system..

Dependencies for VFRJob agent

The VFRJob resource represents the VFR Job. The resource is dependent on the vxfstaskd daemon and vxfsrepld daemon. The vxfstaskd daemon is a scheduler daemon and the vxfsrepld daemon is a replicator daemon.

Refer to the *Cluster Server Administrator's Guide* for more information on resource and group dependencies.

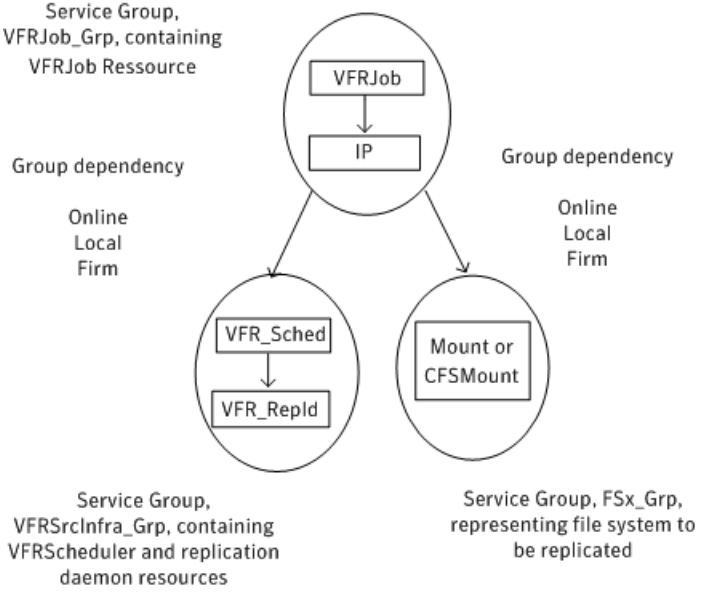
See " Group dependency for VFRJob agent in Custer File Systems or VxFS File Systems" on page 333.

See "Group dependency for VFRJob agent in VxFS File Systems" on page 334.

Group dependency for VFRJob agent in Custer File Systems or VxFS File Systems

For Cluster File System (CFS) or VxFS, configure the VFRJob service group and its dependency as follows:

Figure 8-7 with VFRJob resource in a separate service group than the service group representing VxFS or CFS file system resource



When the VFRJob agent fails for a VxFS file system and you want to failover the VxFS file system, you must configure the group dependency for VFRJob with VxFS file systems.

See "Group dependency for VFRJob agent in VxFS File Systems" on page 334.

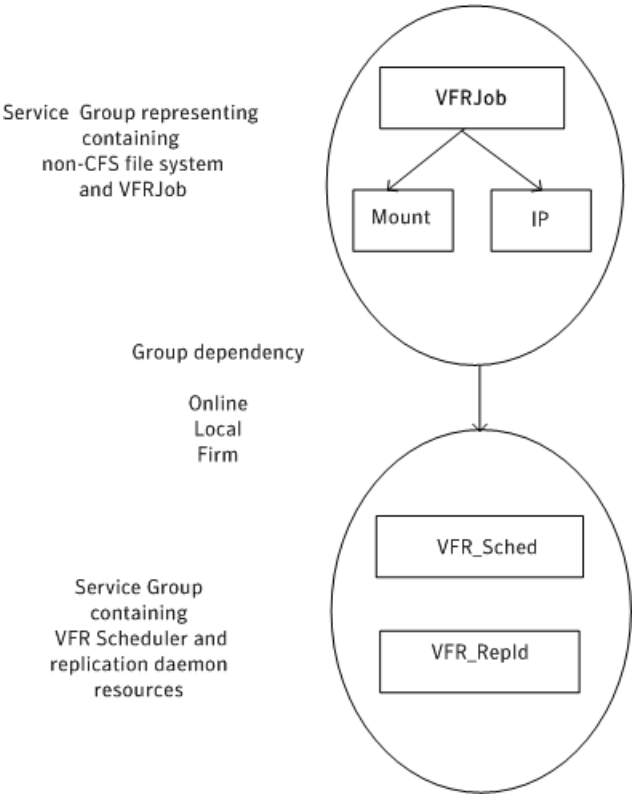
Group dependency for VFRJob agent in VxFS File Systems

Configure the VFRJob and VxFS file system resources as described in this section if you want to failover the file system when the VFRJob faults.

Consider a VxFS file system and VFRJob resource configured in separate service groups. If the VFRJob faults, the VFRJob does not failover to another system. The VFRJob does not failover because the file system is still mounted on the system where the VFRJob faulted.

- Include the VFRJob resource in the service group that represents the file system. Add resource dependency for the VFRJob resource, that is, VFRJob requires Mount resource where the file system is mounted.
- Add group dependency between the service group that contains the VFRJob and the service group that contains the daemon resources. Dependency between the groups must be set to online local firm.

Figure 8-8 VFRJob resource and VxFS file system resource configured in the same service group



High availability of scheduler and replicator daemons

You can configure the vxfstaskd and vxfsrepld daemons which are represented as application type resources in separate service groups of the type parallel.

See "High availability of VFR daemons" on page 337.

Agent functions for VFRJob agent

Table 8-15 Agent functions

Agent function	Description
Online	Starts scheduling of VFRJob.
Offline	Stops scheduling of VFRJob.

Table 8-15 Agent functions *(continued)*

Monitor	Monitors the state of the VFRJob using `vfradmin` command. The agent does not monitor replication.
Clean	Stops scheduling of VFRJob.

State definitions for VFRJob agent

Table 8-16 State definitions

State definition	Description
Online	Indicates Job is running or scheduled to run.
Offline	Indicates that the VFRJob is stopped or is not configured.
Faulted	Indicates VFRJob that was running earlier but is now offline outside VCS control.

Attributes for VFRJob Agent

Table 8-17 Required attributes

Attribute	Description
JobName	Name for the VFRJob. Type and dimension: String Scalar
SrcAddress	IP address of the source system from where the file system is to be replicated. Type and dimension: String Scalar
SrcMountPoint	Mount point on the source system from where the file system is to be replicated. Type and dimension: String Scalar

Resource type definitions for VFRJob agent

The VFRJob agent resource type definition follows:

```
type VFRJob (
   static str ArgList[] = { JobName, SrcMountPoint, SrcAddress }
   str JobName
```

```
    str SrcMountPoint
    str SrcAddress
)
```

High availability of VFR daemons

You can make the vxfstaskd daemon and the vxfsrepld daemon highly available by configuring these daemons as application type resources.

The following configurations are supported:

Configuration of VFRJob service groups on the source system

Consider the following service groups on a source system from where the file system is replicated to the target system:

- Service group FSx_Grp represents a file system (either CVM/CFS or VM/FS)
- Service group VFRJob_Grp represents the VFRJob
- Service group VFRSourceInfra_Grp represents vxfstaskd and vxfsrepld daemons

The group dependencies can be defined as follows:

- VFRJob_Grp requires FSx_Grp online local firm
- VFRJob_Grp requires VFRSrcInfra_Grp online local firm

For more information on resource and group dependencies, refer to the *Cluster Server Administrator's Guide*.

Sample configuration of VFRJob agent on source system

Sample configuration of VFRJob on source system including resource and group dependencies.

```
group VFRJob_Grp (
        SystemList = { sys1= 0, sys2= 1 }
        )

        VFRJob vfr_job (
                JobName = vfr_job_fs1
                SrcMountPoint = "/dgsfs1"
                SrcAddress = " 10.198.95.191"
                RestartLimit = 1
                )
        IP vip_vfr (
```

```
                Device = etho
                Address = "10.198.95.191"
                NetMask = "255.255.255.0"
                )

vfr_job requires vip_vfr
requires group VFRSrcInfra_Grp  online local firm
requires group FSx_Grp online local firm

group VFRSrcInfra_Grp (
        SystemList = {sys1= 0, sys2 = 1}
        AutoStartList = {sys1= 0, sys2 = 1}
        Parallel = 1
        )
Application vfr_repld (
   StartProgram = "/opt/VRTS/bin/vfradmin vxfsrepld start -p 56987"
   StopProgram = "/opt/VRTS/bin/vfradmin vxfsrepld stop"
   CleanProgram = "/opt/VRTSvcs/bin/VFRJob/vfr_repld_clean"
   MonitorProcesses = { "/opt/VRTS/bin/vxfsrepld -p 56987" }
   RestartLimit = 3
   )

Application vfr_sched (
   StartProgram = "/opt/VRTS/bin/vfradmin sched start"
   StopProgram = "/opt/VRTS/bin/vfradmin sched stop"
   CleanProgram = "/opt/VRTSvcs/bin/VFRJob/vfr_sched_clean"
   MonitorProcesses = { "/opt/VRTS/bin/vxfstaskd" }
  RestartLimit = 3
               )
```

Configuration for VFRJob service groups on the target system

(Optional) Configure the service groups on the target system.

Considerations to configure service groups on the target system:

- If the target system is clustered, provide high availability for vxfstaskd and vxfsrepld daemons by placing these daemon resources in a parallel group. For example, in the VFRTargetInfra_Grp service group.
- Service group FSx_Grp2 represents the target file system (either CVM/CFS or VM/FS), where the file system is replicated to from the source system.

- Service group VFRJob_Grp2 represents the VFR job group on the target system. It contains the virtual IP that is used for VFR replication on the target system and VFRJob resource which is disabled (Enabled = 0 is set).
  Note that the VFRJob resource always needs to be disabled on target system. Also, the disabled VFRJob resource on the target system is enabled only when direction of replication needs to be changed for VFR.
  For more details, See "Changing file replication direction" on page 340.
- Define the group dependencies as follows:
  - VFRJob_Grp2 requires FSx_Grp2 in an online local firm dependency.
  - VFRJob_Grp2 requires VFRTargetInfra_Grp in an online local firm dependency.

For more information on resource and group dependencies, refer to the *Cluster Server Administrator's Guide*.

Sample configuration of VFRJob agent on target system

Sample configuration for a target system (under VCS control) to provide high availability for VFR replicator daemon and IP address used by replicator daemon on target systems.

```
group VFRJob_Grp2 (
SystemList = {sysx= 0, sysy = 1}
AutoStartList = {sysx= 0, sysy = 1}
)

VFRJob vfr_job2 (
Enabled = 0
JobName = vfr_job_fs2
SrcMountPoint = "/dgsfs2"
SrcAddress = " 10.198.95.192"
RestartLimit = 1
)

IP vip_vfr2 (
Device = eth0
Address = "10.198.95.192"
NetMask = "255.255.255.0"
)
```

```
vfr_job2 requires vip_vfr2
requires group VFRTargetInfra_Grp online local firm
requires group FSx_Grp2 online local firm
```

Changing file replication direction

If a target file system is written to during a disaster, and then becomes the primary site, the replication direction can be changed so that the old source file system can be made the new target file system.

If both the source and target systems for VFRJob are under VCS control, perform the following steps to change the file replication direction for VFR.

Note: The vfr_job resource represents the VFRJob running on current source system while vfr_job2 resource represents the VFRJob resource that is disabled (Enabled 0) on the target system.

On the source system perform the following steps

1. Offline the VFRJob resource on source system

   ```
   # hares -offline vfr_job -sys source_system,
   ```

   where vfr_job is the VFRJob resource and source_system is the system where VFRJob is online.

2. After the VFRJob goes offline, disable the VFRJob resource.

   ```
   # haconf -makerw

   # hares -modify vfr_job Enabled 0

   # haconf -dump -makero
   ```

3. Perform the steps to change the direction of replication for VFR on both the source system and target system.

   For more details, refer to *Veritas InfoScale Replication Administrator's Guide*.

On the new source system perform the following steps:

After the file replication direction change, the earlier target system becomes the new source system.

1. Enable VFRJob

   ```
   # haconf -makerw

   # hares -modify vfr_job2 Enabled 1

   # haconf -dump -makero
   ```

2. Probe the VFRJob on the new source system

   `# hares -probe vfr_job2 -sys new_source_system` , where new_source_system is system where the virtual IP for VFRJob2 , that is, vfr_vip2 is online.

3. Bring the VFRJob online.

   ```
   # hares -online vfr_job2 -sys new_source_system
   ```

Notes for the VFRJob agent

The notes for VFRJob agent are as follows:

- Veritas recommends using Virtual IP when setting up the VFRJob resource.

Using a different port number for the replication daemon

To run the replication daemon on a different port number perform the following steps:

1. Offline the VFRJob resource.
2. Offline the vfr_repld resource by running the `hares -offline` command.
3. Modify the vfr_repld resource definition for StartProgram and MonitorProcesses with the new port number.
4. Online vfr_repld resource by running the `hares -online` command
5. Online the VFRJob resource by running the `hares -online` command.
6. On the target system skip Step1 and Step5 as VFRJob is disabled.