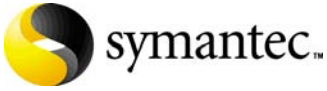
Veritas™ Cluster Server Bundled Agents Reference Guide

ESX

5.1 Maintenance Pack 2



Veritas Cluster Server
Bundled Agents Reference Guide

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

Third-party legal notices

All third-party copyrights associated with this product are listed in the Third Party Copyrights document, which is included on the product disc.

Technical support

For technical assistance, visit http://www.symantec.com/business/support/assistance_care.jsp.

Select phone or email support. Use the Knowledge Base search feature to access resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and our customer email notification service.

Contents

Chapter 4 Virtualization management agents

Chapter 6 Infrastructure and support agents

Chapter 7 Testing agents

Chapter 1

Introduction

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

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

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

Agents can:

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

For a more detailed overview of agents, see the VCS User's Guide.

Resources and their attributes

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

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

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

Modifying agents and their resources

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

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

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

Attributes

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

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

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

Table 1-1 Attribute data types

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

Table 1-1 Attribute data types

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

Table 1-2 Attribute dimensions

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

Chapter 2

Storage agents

This chapter contains:

About the storage agents

Use the storage agents to monitor shared storage.

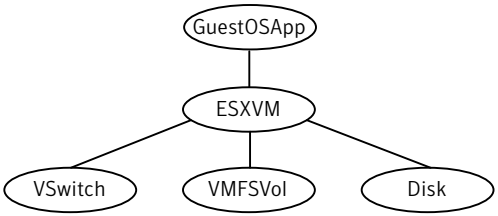
Disk agent

Monitors a disk.

Dependencies

Disk resources have no dependencies. In the following diagram, the ESXVM represents an ESXVirtualMachine resource and the VMFSVol resource represents an VMFSVolume resource.

Figure 2-1 Sample service group for an Disk resource



Using a Disk resource in a service group is further described in the *Veritas Cluster Server Implementation Guide*.

Agent functions

Monitor	Determines if the disk is accessible by performing read I/O operations on the raw disk.

State definitions

FAULTED	Indicates that the disk has unexpectedly stopped working.
UNKNOWN	Indicates that a problem exists either with the configuration or the ability to determine the status of the resource.

Attributes

Table 2-1 Required attributes

Required attribute	Description
Partition	Indicates which partition to monitor. Specify the partition with the full path beginning with a slash (/). If this path is not specified, the name is assumed to reside in /dev/. Type and dimension: string-scalar Example: "/dev/sdc"

Resource type definition

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

Chapter 3

Network agents

This chapter contains:

About the network agents

Use network agents to provide high availability for networking resources.

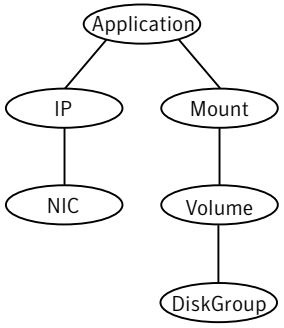
IP agent

The IP agent manages the process of configuring a virtual IP address and its subnet mask on an interface. The interface must be enabled with a physical (or administrative) base IP address before you can assign it a virtual IP address. The virtual IP address must not be in use. You can use this agent when you want to monitor a single IP address on a single adapter.

Dependencies

IP resources depend on NIC resources (except for VCS for ESX).

Figure 3-1 Sample service group for an IP resource



Agent functions

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

State definitions

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

Attributes

Table 3-1 Required attributes

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

Table 3-2 Optional attributes

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

Resource type definition

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

Sample configurations

Configuration 1

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

Configuration using specified NetMask

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

DNS agent

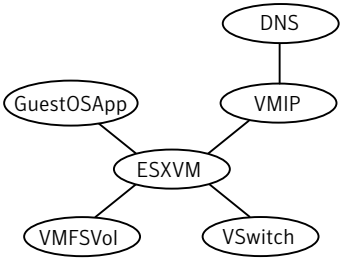
The DNS agent updates and monitors the host name to IP (A record) and the canonical name (CNAME) mapping in the domain name server when failing over virtual machines across subnets. Failing over virtual machines across subnets is also called a wide-area failover.

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

Dependencies

No dependencies exist for the DNS resource.

Figure 3-2 Sample service group for a DNS resource



Agent functions

Online	Queries the authoritative name server of the domain for the A record or the CNAME record and updates the A record or the CNAME record on the name server with the specified alias to canonical name mapping and hostname to IP address mapping. Adds a new A record or CNAME record if a related record is not found. Creates an Online lock file if the Online function was successful.
Offline	Removes the Online lock file, which the Online agent function created.

Monitor	If the Online lock file exists, the Monitor function queries the name servers for the A record or the CNAME record for the alias. It reports back online if the response from at least one of the name servers contains the same canonical name associated with the alias in the Hostname attribute. If no servers return the appropriate name, the monitor reports the resource as offline.
Clean	Removes the Online lock file, if it exists.
Open	Removes the Online lock file if the Online lock file exists, and the A record or the CNAME record on the name server does not contain the expected alias or canonical name mapping hostname to IP address.

State definitions

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

Attributes

Table 3-3 Required attributes

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

Table 3-4 Optional attributes

Optional attribute	Description
Alias	You must set either the Alias or the IPAddress attribute or both. A string representing the alias to the canonical name. Type and dimension: string-scalar Example: "www" Where www is the alias to the canonical name mtv.symantec.com.

Table 3-4 Optional attributes

Optional attribute	Description
IPAddress	You must set either the IPAddress or the Alias attribute or both. Specifies the IP address assigned to the hostname. Type and dimension: string-scalar Example: "10.123.12.55"
StealthMasters	The list of primary master name servers in the domain. Optional if the zone's name server record lists the primary master name server. If the primary master name server is a stealth server, define this attribute. A stealth server is a name server that is authoritative for a zone, but is not listed in that zone's name server records. Type and dimension: string-keylist Example: { "10.190.112.23" }
TSIGKeyFile	Required when you configure DNS for secure updates. Specifies the absolute path to the file containing the private TSIG (Transaction Signature) key. Type and dimension: string-scalar Example: "/var/tsig/Ksymantec.com.+157+00000.private"

Resource type definition

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

Online queries

The Online function performs different kinds of queries depending on whether you specify the Alias or IPAddress attributes, or if you specify both.

If you specify both the Alias and IPAddress attributes both of the following sections are true:

- "Setting the Alias attribute" on page 28
- "Setting the IPAddress attribute" on page 28

Setting the Alias attribute

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

Setting the IPAddress attribute

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

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

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

Monitor scenarios

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

Table 3-5 Monitor scenarios for the Online lock file

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

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

Sample web server configuration

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

Sample DNS configuration

```
DNS www (
        Domain = "example.com"
        Alias = www
        Hostname = virtual1
        )
```

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

- ESX

```
www   CNAME   wwwvirtual1
```

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

DNS agent prerequisites

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

Secure DNS update

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

Setting up secure updates using TSIG keys

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

To use secure updates using TSIG keys

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

   ```
   # dnssec-keygen -a HMAC-MD5 -b 512 -n HOST symantec.com.
       Ksymantec.com.+157+00000
   ```

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

   ```
   # cat Ksymantec.com.+157+00000.key
       symantec.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 symantec.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 symantec.com. ; } ;
   ```

6. Save and restart the named process.

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

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

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

Chapter 4

Virtualization management agents

This chapter contains:

About the virtualization management agents

Virtualization management agents allow you to manage certain virtualized environments and maintain their high availability.

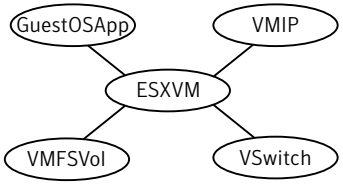
ESXVirtualMachine agent

The ESXVirtualMachine agent brings online, takes offline, and monitors virtual machines that are configured on the ESX Server. You must use this agent when you want to monitor virtual machines, and make them highly available. The ESXVirtualMachine agent detects when an operating system crashes, and initiates the node failover.

Dependencies

This resource depends on the VMFSVolume and the VSwitch resources for its datastore and network.

Figure 4-1 Sample service group for an ESXVirtualMachine resource, where ESXVM stands for an ESXVirtualMachine resource and VMFSVol stands for a VMFSVolume resource



Requirements

This agent requires VMware Tools to operate. You must also make sure that the guestinfo interfaces of the VMware Virtual Machine tools are enabled. Note that these interfaces are enabled by default.

Agent functions

Online	Registers the virtual machine if it is not already registered. Starts the virtual machine. The Online agent function fails with an error message if the ESX Server host is in maintenance mode.
Offline	Attempts a graceful shut down of the virtual machine.
Monitor	Detects the virtual machine's state.

Action	■ testVCConnect This agent function verifies that you can connect to the VirtualCenter server with the current values of the attributes for this resource. See "Verifying if a service group can be migrated" on page 39. ■ migrate This agent function helps to migrate the virtual machine from one node to another. This action script is for internal use only, and Symantec recommends that you do not use it. For migrating virtual machines, use the `hagrp -migrate` command. For more information, see the *Veritas Cluster Server for VMware ESX Implementation Guide*.
Clean	Forcefully shuts down the virtual machine.
Close	Cleans up internal agent datastructures.

State definitions

ONLINE	Indicates that the virtual machine is up and has a heartbeat.
OFFLINE	Indicates that the virtual machine is up but is not sending a heartbeat, or is down but is still registered on the ESX Server.
FAULTED	Indicates that the virtual machine is up but not sending a heartbeat, or is down but still registered on the ESX Server.
UNKNOWN	Indicates the agent cannot determine the virtual machine's state.

Attributes

Table 4-1 Required attribute

Required attribute	Description
CfgFile	Specifies the complete pathname of the configuration file for a virtual machine. The pathname starts with the slash (/) preceding the file name. Do not use the alias. Use the UUID path. When you configure a virtual machine for high availability using the Veritas Virtualization Manager, the value of this attribute is set automatically. Type and dimension: string-scalar Example: /vmfs/volumes/5c5d8e06-da11f7ce-7892-930b00a53cd2/sles9v2/sles9v2.vmx

Table 4-2 Optional attributes

Optional attribute	Description
MonitorHB	This flag enables heartbeat monitoring for a virtual machine. When the value of this attribute is 1, the agent detects a heartbeat failure of the virtual machine, and flags it as a fault to VCS. The virtual machine must have VMware Tools installed and running inside of the virtual machine. Type and dimension: boolean-scalar Default: "1"
username	The username to connect to the VirtualCenter Server. Migrate functionality requires that you set this attribute. When you configure a virtual machine for high availability using the Veritas Virtualization Manager, the value of this attribute is set automatically. Type and dimension: string-scalar

Table 4-2 Optional attributes

Optional attribute	Description
password	The password to connect to the VirtualCenter Server. This password must be encrypted. For encryption, use the vcsencrypt utility. For more information on using the vcsencrypt utility, see the *Veritas Cluster Server User's Guide*. Migrate functionality requires that you set this attribute. When you configure a virtual machine for high availability using the Veritas Virtualization Manager, the value of this attribute is set automatically. Type and dimension: string-scalar
sslcert	The path to the keystore file for the VirtualCenter Server. Migrate functionality requires that you set this attribute. You must copy the keystore file to each node in the cluster. Use the same path on each node to save time. Set the value of the sslcert attribute to the full pathname of the keystore file on each VCS node. The value of this attribute is **not** set automatically by the Veritas Virtualization Manager. For more information on creating and using keystore files, see the *Veritas Cluster Server for VMware ESX Implementation Guide*. Type and dimension: string-scalar
esxhostdomain	The domain name of the ESX host, from the VirtualCenter Server's perspective. If you are unsure, check the ESX Server names when connected to the VirtualCenter Server through your preferred client. Migrate functionality requires that you set this attribute. When you configure a virtual machine for high availability using the Veritas Virtualization Manager, the value of this attribute is set automatically. Type and dimension: string-scalar

Table 4-2 Optional attributes

Optional attribute	Description
vmname	The display name of this virtual machine. If you have set your resource name to be the same as the display name, skip this attribute. Migrate functionality requires that you set this attribute. When you configure a virtual machine for high availability using the Veritas Virtualization Manager, the value of this attribute is set automatically. Type and dimension: string-scalar
VCserver	The name or the IP address of the Virtual Infrastructure Server. When you configure a virtual machine for high availability using the Veritas Virtualization Manager, the value of this attribute is set automatically. Type and dimension: string-scalar

Resource type definition

```
type ESXVirtualMachine (
    static int IntentionalOffline = 1
    static keylist SupportedActions = { growfs, migrate }
    static boolean Migratable = 1
    static keylist ExternalStateChange = { OnlineGroup,
    OfflineGroup }
    static str ArgList[] = { CfgFile, MonitorHB, VCserver, username,
    password, sslcert, esxhostdomain, vmname }
    str CfgFile
    boolean MonitorHB = 1
    str VCserver
    str username
    str password
    str sslcert
    str esxhostdomain
    str vmname
    )
```

Verifying if a service group can be migrated

You can use the testVCConnect agent function to verify that the ESXVirtualMachine resource attributes are correctly configured for connecting to the VirtualCenter server.

Run the testVCConnect script on each node of the cluster, if you plan to use hagrp -migrate functionality. You can find the script in the actions directory /opt/VRTSvcs/bin/ESXVirtualMachine/actions/

Run the testVCConnect function:

```
# hares -action resname token -sys vcssystemname
```

The following line is an example:

```
# hares -action evm testVCConnect -sys esxNode1
```

Where evm is the name of the ESXVirtualMachine resource, testVCConnect is the token name, and esxNode1 is the name of the node from where you want to test the connection.

For more information on `hagrp -migrate` functionality, see the:

- *Veritas Cluster Server Implementation Guide for ESX*

Sample configurations

Basic Suse configuration

```
ESXVirtualMachine sles9vm3 (
    CfgFile = "/vmfs/volumes/44a15b88-f73de898-a4c3-00093d10858b/
    vm3/vm3.vmx"
    )
```

ESXVirtualMachine agent configured for migration

```
ESXVirtualMachine vm3 (
    CfgFile = "/vmfs/volumes/44a15b88-f73de898-a4c3-00093d10858b/
    vm3/vm3.vmx"
    VCserver = pc54
    username = Administrator
    password = dllLe
    sslcert = "/root/vmware-certs/pc54.keystore"
    esxhostdomain = "symantec.com"
    vmname = vm3
    )
```

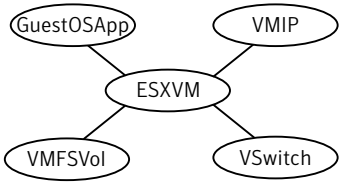
GuestOSApp agent

Reflects the state of an application that runs in a guest operating system.

Dependencies

This resource depends on the ESXVirtualMachine resource.

Figure 4-2 Sample service group for a GuestOSApp resource, where ESXVM stands for an ESXVirtualMachine resource and VMFSVol stands for a VMFSVolume resource



Requirements

This agent requires VMware Tools to operate. You must also make sure that the guestinfo interfaces of the VMware Virtual Machine tools are enabled.

Agent functions

Action — The action function tests if the GuestOSApp resource is properly configured. The Action functions that are implemented can return:

- fault
  VCS uses this agent function internally.
- getappstate
  This agent function returns the state of the application that runs inside a virtual machine as set in the VMWare guestinfo interface. Typically, you can use this action to verify if a GuestOSApp resource (and a corresponding Application resource inside the virtual machine) are properly configured and mapped. The action script takes no parameters and can be run using your preferred VCS management interface. If you use a command line interface, run the command:
  `hares -action` *`resname`* `getappstate -sys` *`vcssystemname`*

Monitor — Probes the state of the application through the VMware ESX virtual machine guestinfo variable and reflects its state accordingly.

State definitions

ONLINE — If the VMware ESX virtual machine guestinfo variable state suggests that the application is ONLINE, the agent returns an ONLINE state.

FAULTED — If the VMware ESX virtual machine guestinfo variable state suggests that the application is OFFLINE, the agent returns an FAULTED state.

Attributes

Table 4-3 Required attribute

Required attribute	Description
VMwareResName	Use the exact name as it appears for the ESXVirtualMachine resource in the VCS configuration. Type and dimension: string-scalar

Resource type definition

```
type GuestOSApp (
    static int IntentionalOffline = 1
    static int MonitorInterval = 60
    static int OnlineWaitLimit = 5
    static keylist SupportedActions = { "fault"}
    static str ArgList[] = { "VMwareResName:CfgFile" }
    str VMwareResName
    )
```

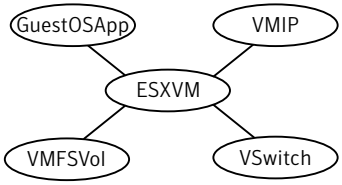
VMFSVolume agent

The VMFSVolume agent monitors the VMFS volumes on an ESX Server.

Dependencies

No dependencies exist for the VMFSVolume resource.

Figure 4-3 Sample service group for a VMFSVolume resource, where ESXVM stands for an ESXVirtualMachine resource and VMFSVol stands for a VMFSVolume resource



Agent functions

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

State definitions

ONLINE — Defines a mounted and accessible data store on the ESX host.

Attributes

Table 4-4 Required attribute

Required attribute	Description
Volume	Specifies the volume-UID of the VMFS datastore that you want to monitor. The agent returns online if the datastore is mounted and accessible on the ESX host. Type and dimension: string-scalar Example: /vmfs/volumes/5c5d8e06-da11f7ce-7892-930b00a53cd2

Resource type definition

```
type VMFSVolume (
    static int MonitorInterval = 30
    static str ArgList[] = { Volume }
    str Volume[]
    )
```

Sample configuration

```
VMFSVolume Shared3 (
    Volume = "/vmfs/volumes/5c5d8e06-da11f7ce-7892-930b00a53cd2"
    )
```

VMIP agent

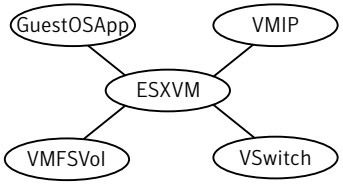
The virtual machine IP (VMIP) agent updates the IP address, subnet mask, gateway, and DNS on an interface in a virtual machine.

You cannot dynamically change the VMIP resource's attribute values. If the VMIP resource is in an ONLINE state, you need to take the resource offline, update the attribute values, and then bring the resource back online.

Dependencies

The VMIP resource depends on the ESXVirtualMachine resource.

Figure 4-4 Sample service group for a VMIP resource, where ESXVM stands for an ESXVirtualMachine resource and VMFSVol stands for a VMFSVolume resource



Requirements

This agent requires VMware Tools to operate. You must also make sure that the guestinfo interfaces of the VMware Virtual Machine tools are enabled.

Agent functions

Online	Configures the IP address, subnet mask, gateway, and the DNS to the virtual NIC. Creates an Online lock file if the Online function was successful.
Offline	Removes the Online lock file, which the Online agent function creates.
Monitor	Checks if the Online lock file exists. Reports back in an ONLINE state if the file exists.
Clean	Removes the Online lock file.
Open	Removes the Online lock file.

State definitions

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

Attributes

Table 4-5 Required attribute

Required attribute	Description
VMwareResName	The name of the VCS resource that manages the virtual machine. Type and dimension: string-scalar
IPAddress	The IP address that is assigned to the virtual machine interface. Type and dimension: string-scalar Example: 10. 123.12.55
MACAddress	The MAC address of the virtual NIC, which must be the MAC address that the ESX Server assigns to the virtual NIC that is inside the virtual machine. Type and dimension: string-scalar Example: 00:0C:29:FF:86:2E
NetMask	The subnet mask that is associated with the IP address. You must specify this value in decimal (base 10). Type and dimension: string-scalar Example 255.255.255.0
Gateway	The default gateway for the virtual machine. Type and dimension: string-scalar Example: 10.123.17.1
DNS	List of DNS servers in the required search order. Type and dimension: string-keylist Example: {10.123.17.160 , 10.123.17.161}

Table 4-6 Optional attribute

Required attribute	Description
NicConf	The NicConf attribute helps support configuration of multiple network interfaces using a single resource of type VMIP. The updated attribute type takes the MAC address of the NIC as the key value and the IP address as the data. You can append a nonstandard netmask in decimal notation to the IP address with a ":" as separator. Type and dimension: string-scalar Example: 00:50:56:94:06:5D = 10.100.90.16:255.255.248.0, 00:50:56:94:64:B1 = 192.168.1.18

Resource type definition

```
type VMIP (
    static int MonitorInterval = 300
    static str ArgList[] = { "VMwareResName:CfgFile", IPAddress,
    MACAddress, NetMask, Gateway, DNS, NICConf }
    str VMwareResName
    str IPAddress
    str MACAddress
    str NetMask
    str Gateway
    str DNS[]
    str NICConf{}
)
```

Sample configuration

```
VMIP vmIP_rhel4_32bit_vm (
    VMwareResName = esxVM_rhel4_32bit_vm
    IPAddress = "10.100.90.18"
    MACAddress = "00:50:56:94:57:05"
    NetMask = "255.255.248.0"
    Gateway = "10.100.88.1"
    DNS = { "10.150.88.20", "191.158.1.3" }
    NICConf {
    "00:50:56:94:06:5D " = "10.100.90.16:255.255.248.0",
    "00:50:56:94:64:B1" = "192.168.1.18" }
)
```

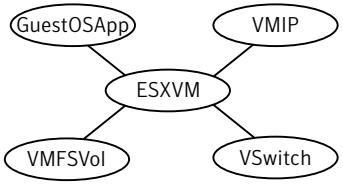
VSwitch agent

The VSwitch agent monitors a virtual switch on an ESX Server.

Dependencies

No dependencies exist for the VSwitch resource.

Figure 4-5 Sample service group for a VSwitch resource, where ESXVM stands for an ESXVirtualMachine resource and VMFSVol stands for a VMFSVolume resource



Agent functions

Monitor	The agent performs a hardware link test using the mii-tool, and returns online if any of the switch's uplinks are up.

State definitions

ONLINE	Indicates that the virtual switch is working.
FAULTED	Indicates that the virtual switch has failed.
UNKNOWN	Indicates the agent cannot determine the virtual switch's interface states. This state may be due to an incorrect configuration.

Attribute

Table 4-7 Required attribute

Required attribute	Description
VirtualSwitch	Specifies the name of the virtual switch that you want to monitor. An ESXVirtualMachine resource that depends on a VSwitch resource, requires that all the VSwitch resource names be identical across all the cluster nodes. Note that VCS automatically makes sure that the names are identical when you use the Veritas Virtualization Manager. Type and dimension: string-scalar Example: "vSwitch0"

Resource type definition

```
type VSwitch (
    static int MonitorInterval = 30
    static str ArgList[] = { VirtualSwitch }
    static str Operations = None
    str VirtualSwitch
    )
```

Sample configuration

```
VSwitch vSwitch1 (
    VirtualSwitch = vSwitch0
    )
```

ESXHost agent

The ESXHost agent, an internal-use only agent, enforces compatibility between VCS and VMware's DRS and maintenance modes.

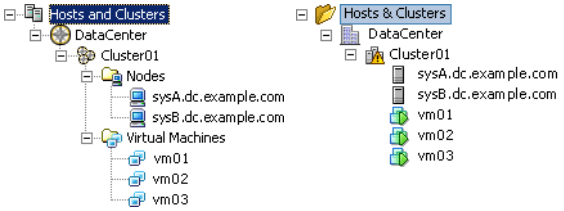
The ESXHost agent is automatically deployed on each node in the cluster, and is created in a service group called the ESXHostServiceGroup. This is a standalone, parallel service group, which you should not modify. The ESXHost agent, and its corresponding service group, are required on each node of your VCS for VMware ESX cluster.

The ESXHost agent enforces a two-way compatibility between VCS and VMware's DRS and maintenance mode functions. If you bring down a node with VMware maintenance mode to perform system maintenance tasks, the ESXHost agent evacuates all VCS service groups to another ESX node in the VCS cluster and performs a system freeze for the evacuated node. The agent also provides reverse compatibility—if it detects that a VCS system is frozen, and no active virtual machines are on the host, it brings down the host to VMware maintenance mode.

Prerequisites

For the ESXHost agent to work properly, ensure that an exact match exists between the overlapping nodes on VCS and VMware clusters. Make sure that the DRS cluster uses exactly the same hosts as the hosts that overlap with the VCS cluster.

Figure 4-6 Maintaining an exact correlation between products



VCS cluster VMware cluster

Agent functions

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

Attribute

Table 4-8 Internal attributes

Required attribute	Description
OfflineMonitorInterval	For internal use only. Do not modify.

Resource type definition

```
type ESXHost (
    static int OfflineMonitorInterval = 60
    static str Operations = None
)
```

Chapter 5

Service and application agents

This chapter contains:

About the service and application agents

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

Application agent

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

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

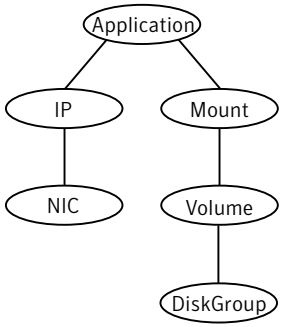
You can monitor the application in the following ways:

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

Dependencies

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

Figure 5-1 Sample service group for an Application resource



Agent functions

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

Offline	Runs the StopProgram attribute with the specified parameters in the context of the specified user.
Monitor	If you specify the MonitorProgram attribute, the agent executes the user-defined MonitorProgram in the user-specified context. If you specify the PidFiles attribute, the routine verifies that the process ID that is found in each listed file is running. If you specify the MonitorProcesses attribute, the routine verifies that each listed process is running in the context you specify. Use any combination among these attributes (MonitorProgram, PidFiles, or MonitorProcesses) to monitor the application. If any one the processes that are specified in either PidFiles or MonitorProcesses is determined not to be running, the monitor returns OFFLINE. If the process terminates ungracefully, the monitor returns OFFLINE and failover occurs.
Clean	Terminates processes specified in PidFiles or MonitorProcesses. Ensures that only those processes (that are specified in the MonitorProcesses attribute) running with the user ID specified in the User attribute are killed. If the CleanProgram is defined, the agent executes the CleanProgram.

State definitions

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

Attributes

Table 5-1 Required attributes

Required attribute	Description
StartProgram	The executable, created locally on each node, which starts the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. **Note:** Do not use the opening and closing ({ }) brace symbols in this string. Type and dimension: string-scalar Example: "/usr/sbin/samba start"
StopProgram	The executable, created locally on each node, which stops the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. **Note:** Do not use the opening and closing ({ }) brace symbols in this string. Type and dimension: string-scalar Example: "/usr/sbin/sample_app stop"
At least one of the following attributes: ■ MonitorProcesses ■ MonitorProgram ■ PidFiles	See “Optional attributes” on page 57.

Table 5-2 Optional attributes

Optional attribute	Description
CleanProgram	The executable, created locally on each node, which forcibly stops the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. Type and dimension: string-scalar
MonitorProcesses	A list of processes that you want monitored and cleaned. Each process name is the name of an executable. Qualify the executable name with its complete path if the path starts the executable. The process name must be the name that the `ps -ef` command displays for the process. Type and dimension: string-vector Example: "nmbd"
MonitorProgram	The executable, created locally on each node, which monitors the application. Specify the complete path of the executable. Applicable command line arguments follow the name of the executable and have spaces separating them. MonitorProgram can return the following VCSAgResState values: OFFLINE value is 100; ONLINE values range from 101 to 110 (depending on the confidence level); 110 equals confidence level of 100%. Any other value = UNKNOWN. **Note:** Do not use the opening and closing ({ }) brace symbols in this string. Type and dimension: string-scalar

Table 5-2 Optional attributes

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

Resource type definition

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

Sample configurations

Configuration 1

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

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

Configuration 2

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

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

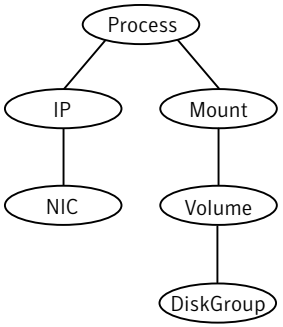
Process agent

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

Dependencies

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

Figure 5-2 Sample service group for a Process resource



Agent functions

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

State definitions

ONLINE	Indicates that the specified process is running in the specified user context.

OFFLINE	Indicates that the specified process is not running in the specified user context.
FAULTED	Indicates that the process has terminated unexpectedly.
UNKNOWN	Indicates that the agent can not determine the state of the process.

Attributes

Table 5-3 Required attribute

Required attribute	Description
PathName	Complete pathname to access an executable program. This path includes the program name. If a script controls the process, the PathName defines the complete path to the shell. This attribute must not exceed 256 characters. Type and dimension: string-scalar Example: "/usr/sbin/proc1"

Table 5-4 Optional attributes

Optional attribute	Description
Arguments	Passes arguments to the process. If a script controls the process, the script is passed as an argument. Separate multiple arguments with a single space. A string cannot accommodate more than one space between arguments, nor allow for leading or trailing whitespace characters. Type and dimension: string-scalar

Table 5-4 Optional attributes

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

Resource type definition

```
type Process (
    static keylist SupportedActions = { "program.vfd", getcksum }
    static str ArgList[] = { PathName, Arguments, UserName,
    Priority, PidFile }
    str PathName
    str Arguments
    str UserName = root
    str Priority = 10
    str PidFile
)
```

Sample configurations

Configuration

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

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

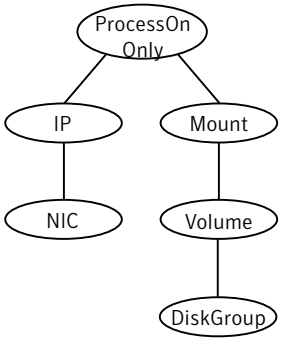
ProcessOnOnly agent

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

Dependencies

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

Figure 5-3 Sample service group for a ProcessOnOnly 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.
FAULTED	Indicates that the process has unexpectedly terminated.
UNKNOWN	Indicates that the agent can not determine the state of the process.

Attributes

Table 5-5 Required attributes

Required attribute	Description
PathName	Defines complete pathname to access an executable program. This path includes the program name. If a process is controlled by a script, the PathName defines the complete path to the shell. The PathName attribute must not exceed 256 characters. Type and dimension: string-scalar

Table 5-6 Optional attributes

Optional attribute	Description
Arguments	Passes arguments to the process. If a process is controlled by a script, the script is passed as an argument. Multiple arguments must be separated by a single space. A string cannot accommodate more than one space between arguments, nor allow for leading or trailing whitespace characters. Type and dimension: string-scalar Example: "-bd -q30m"
IgnoreArgs	A flag that indicates whether monitor ignores the argument list. ■ If the value is 0, it checks the process pathname and argument list. ■ If the value is 1, it only checks for the executable pathname and ignores the rest of the argument list. Type and dimension: boolean-scalar Default: 0

Table 5-6 Optional attributes

Optional attribute	Description
PidFile	The file that contains the process ID for the monitoring process. Specify the PidFile attribute for the monitoring process to use the Pid. Otherwise, to complete the monitoring process the agent uses the ps output. Note that when you use scripts, or other indirect mechanisms, to start processes, you must set the PidFile attribute 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
)
```

Sample configurations

Configuration 1

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

Configuration 2

```
include "types.cf"

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

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

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

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

Chapter 6

Infrastructure and support agents

This chapter contains:

About the infrastructure and support agents

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

NotifierMngr agent

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

You cannot dynamically change the attributes of the NotifierMngr agent using the `hares -modify` command. Changes made using this command are 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=xyz &
```

Dependency

The NotifierMngr resource depends on the NIC resource.

Agent functions

Online	Starts the notifier process with its required arguments.
Offline	VCS sends a `SIGABORT`. If the process does not exit within one second, VCS sends a `SIGKILL`.
Monitor	Monitors the notifier process.
Clean	Sends `SIGKILL`.

State definitions

ONLINE	Indicates that the Notifier process is running.
OFFLINE	Indicates that the Notifier process is not running.
UNKNOWN	Indicates that the user did not specify the required attribute for the resource.

Attributes

Table 6-1 Required attributes

Required attribute	Description
SnmpConsoles	Specifies the machine name of the SNMP manager and the severity level of the messages to be delivered to the SNMP manager. The severity levels of messages are `Information`, `Warning`, `Error`, and `SevereError`. Specifying a given severity level for messages generates delivery of all messages of equal or higher severity. SnmpConsoles is a required attribute if SmtpServer is not specified; otherwise, SnmpConsoles is an optional attribute. Specify both SnmpConsoles and SmtpServer if desired. Type and dimension: string-association Example: "172.29.10.89" = Error, "172.29.10.56" = Information
SmtpServer	Specifies the machine name of the SMTP server. SmtpServer is a required attribute if SnmpConsoles is not specified; otherwise, SmtpServer is an optional attribute. You can specify both SmtpServer and SnmpConsoles if desired. Type and dimension: string-scalar Example: "smtp.example.com"

Table 6-2 Optional attributes

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

Table 6-2 Optional attributes

Optional attribute	Description
MessagesQueue	Size of the VCS engine's message queue. Minimum value is 30. Type and dimension: integer-scalar Default: 30
NotifierListeningPort	Any valid, unused TCP/IP port numbers. Type and dimension: integer-scalar Default: 14144
SmtpFromPath	Set to a valid email address, if you want the notifier to use a custom email address in the FROM: field. Type and dimension: string-scalar Example: "usera@example.com"
SmtpRecipients	Specifies the email address where SMTP sends information and the severity level of the messages. The severity levels of messages are Information, Warning, Error, and SevereError. Specifying a given severity level for messages indicates that all messages of equal or higher severity are received. **Note:** SmtpRecipients is a required attribute if you specify SmtpServer. Type and dimension: string-association Example: "james@symantec.com" = SevereError, "admin@symantec.com" = Warning

Table 6-2 Optional attributes

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

Table 6-2 Optional attributes

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

Resource type definition

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

Sample configuration

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

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

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

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

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

Configuration

```
system north

system south

group  NicGrp (
    SystemList = { north, south}
    AutoStartList = { north }
    Parallel = 1
    )

    Phantom my_phantom (
    )

    NIC    NicGrp_eth0 (
        Enabled = 1
        Device  = eth0
        )

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

    Proxy nicproxy(
    TargetResName = "NicGrp_eth0"
```

```
)

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

ntfr requires nicproxy

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

VRTSWebApp agent

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

Agent functions

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

State definitions

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

Attributes

Table 6-3 Required attributes

Required attribute	Description
AppName	Name of the application as it appears in the Web server. Type and dimension: string-scalar Example: "cmc"

Table 6-3 Required attributes

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

Resource type definition

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

Sample configuration

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

Proxy agent

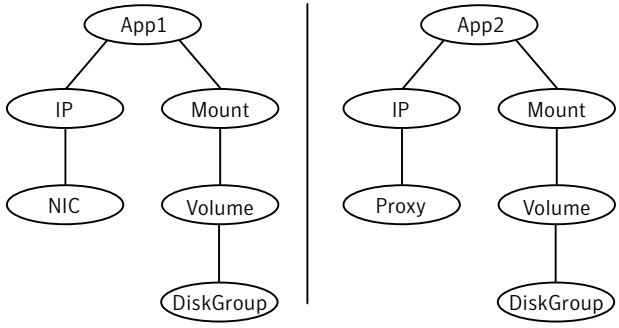
The Proxy agent mirrors the state of another resource on a local or remote system. It provides a means to specify and modify one resource and have its state reflected by its proxies. You can use the agent when you need to replicate the status of a resource.

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

Dependencies

No dependencies exist for the Proxy resource.

Figure 6-1 Sample service group for an Proxy resource



Agent functions

Monitor Determines status based on the target resource status.

Attributes

Table 6-4 Required attribute

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

Table 6-5 Optional attribute

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

Resource type definition

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

Sample configurations

Configuration 1

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

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

Configuration 2

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

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

Configuration 3

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

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

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

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

    ip1 requires mnic


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

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

Phantom agent

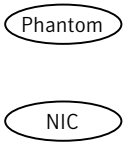
The Phantom agent enables VCS to determine the status of parallel service groups that do not include OnOff resources. Do not use the Phantom agent in failover service groups. You can use the agent to determines the state of service groups having resources of type None only.

Do not attempt manual online or offline operations on the Phantom resource or on the service group containing the Phantom resource. Doing so may result in upredictable behavior.

Dependencies

No dependencies exist for the Phantom resource.

Figure 6-2 Sample service group for a Phantom resource



Agent functions

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

Resource type definition

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

Sample configurations

Configuration 1

```
Phantom (
)
```

Configuration 2

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

```
include "types.cf"

cluster PhantomCluster

system sysa

system sysb

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

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

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

Chapter 7

Testing agents

This chapter contains:

About the program support agents

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

ElifNone agent

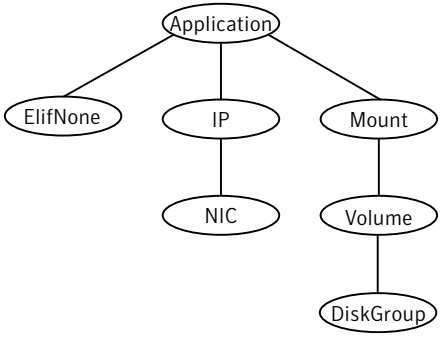
The ElifNone agent monitors a file. It checks for the file's absence.

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

Dependencies

No dependencies exist for the ElifNone resource.

Figure 7-1 Sample service group for an ElifNone resource



Agent function

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

Attributes

Table 7-1 Required attribute

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

Resource type definition

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

Sample configuration

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

FileNone agent

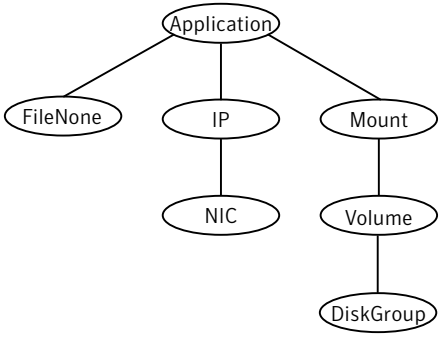
Monitors a file—checks for the file's existence.

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

Dependencies

No dependencies exist for the FileNone resource.

Figure 7-2 Sample service group for an FileNone resource



Agent functions

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

Attribute

Table 7-2 Required attribute

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

Resource type definition

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

Sample configuration

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

FileOnOff agent

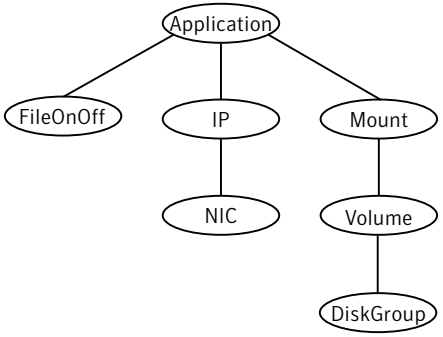
The FileOnOff agent creates, removes, and monitors files.

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

Dependencies

No dependencies exist for the FileOnOff resource.

Figure 7-3 Sample service group for a FileOnOff resource



Agent functions

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

Attribute

Table 7-3 Required attribute

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

Resource type definition

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

Sample configuration

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

FileOnOnly agent

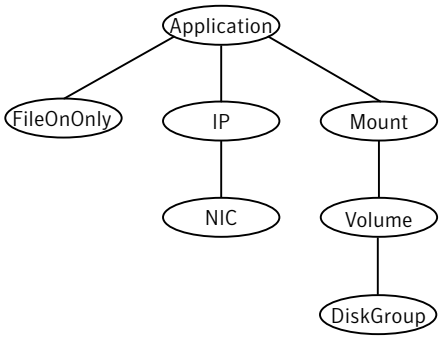
The FileOnOnly agent creates and monitors files.

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

Dependencies

No dependencies exist for the FileOnOnly resource.

Figure 7-4 Sample service group for a FileOnOnly resource



Agent functions

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

Attribute

Table 7-4 Required attributes

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

Resource type definition

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

Sample configuration

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

Glossary

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

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

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

entry point
See agent function.

floating IP address
See virtual IP address.

logical IP address
Any IP address assigned to a NIC.

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

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

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

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

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

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

test IP address

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

virtual IP address

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

Index

C

D

E

F

G

I

K

M

N

O

P

R

S

T

V