TECHNICAL BRIEF: STORAGE MANAGEMENT

Migrating HP Serviceguard for Linux to Veritas Cluster Server for Linux

Technical Brief:

Migrating HP Serviceguard for Linux to Veritas Cluster Server for Linux

Contents

Executive Summary **1**

Third-party legal notices 1

Licensing and registration 1

Technical support 1

Introduction **1**

Audience 1

Technology Overview **2**

Veritas Cluster Server (VCS) for Linux 2

Veritas Cluster Server Architecture Introduction **2**

Veritas Cluster Server clustering concepts 2

Cluster Communication 6

Term Comparison - SGLX and VCS 9

Command Comparison - SGLX and VCS 9

Veritas Cluster Server cluster heartbeats 10

Migration Procedure **10**

Planning Phase 10

Identify applications and resources under Serviceguard for Linux control 10

Veritas Cluster Server Hardware Prerequisites 11

Veritas Cluster Server Linux OS Prerequisites 12

Preparing for Veritas Cluster Server Installation 12

Implementation Phase 12

Shutdown the SGLX cluster 12

Uninstall SGLX software 12

Veritas Cluster Server installation 13

Veritas Cluster Server Heartbeats 13

Veritas Cluster Server Cluster Creation 13

Verify the Veritas Cluster Server Cluster 13

Veritas Cluster Server validation and testing 14

Serviceguard for Linux to VCS Flowchart 14
Methods of Controlling the Veritas Cluster Server Cluster 14
Summary of VCS management capabilities 15
VCS Command Line Interface (CLI) 15
VCS Java Console 15
VCS Cluster Simulator. 15
VCS Management Console 15
Storage Foundation Manager. 16
Veritas Operations Manager 16
Appendix reference information 17
Migration Planning – VCS Cluster Information 17
Step-by-step migration with sample applications – SGLX -> VCS 22
SGLX Configuration Files Examples 24
Veritas Cluster Server Configuration Files Examples 25
Serviceguard and Veritas Cluster Server Configuration Files Migration Example 29
Reference Documentation 73

Executive Summary

This white paper, created with the assistance of HP, illustrates a process to migrate an HP Serviceguard for Linux (SGLX) cluster to Veritas Cluster Server (VCS). An introduction to the architecture of VCS is described including sections comparing SGLX and VCS which contrast cluster terminology and describe architecture differences. A step-by-step process describes how to use configuration information from an existing SGLX cluster to quickly migrate to a VCS cluster with similar functionality.

Third-party legal notices

Third-party software may be recommended, distributed, embedded, or bundled with this Veritas product. Such third-party software is licensed separately by its copyright holder. All third-party copyrights associated with this product are listed in the Veritas Cluster Server Release Notes.

Licensing and registration

Veritas Cluster Server is a licensed product. See the Veritas Cluster Server Installation Guide for license installation instructions.

Technical support

For technical assistance, visit:

http://www.symantec.com/enterprise/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.

Introduction

This document is intended to provide information to assist with the migration of a cluster from Serviceguard for Linux (SGLX) to Veritas Cluster Server (VCS) for Linux. HP Serviceguard on Linux is being discontinued and with this document it is our intention to illustrate the migration path to Veritas Cluster Server as an alternative to Serviceguard for Linux. Please review product documentation before installing VCS.

It is a best practice to validate that the current Serviceguard cluster configuration behaves as expected to ensure that the environment is in a known good state before beginning a migration.

Audience

This document is targeted for technical users of HP Serviceguard for Linux who wish to migrate to Veritas Cluster Server on Linux. It is assumed that the reader has a general understanding of HP Serviceguard for Linux, the Linux Operating System and Veritas Cluster Server. For more information, see http://www.hp.com/go/sglx for Serviceguard for Linux, the Linux OS vendor's website and for Veritas Cluster Server on Linux see http://www.symantec.com/business/cluster-server.

Technology Overview

Veritas Cluster Server (VCS) for Linux

Veritas Cluster Server from Symantec connects multiple, independent systems into a management framework for increased availability. Each system, or node, runs its own operating system and cooperates at the software level to form a cluster. These systems can be either a physical or virtual server. VCS links commodity hardware with intelligent software to provide application failover and control. When a node or a monitored application fails, other nodes can take predefined actions to take over and bring up services elsewhere in the cluster.

Veritas Cluster Server is the industry's leading clustering solution for reducing business critical applications' planned and unplanned downtime. VCS can detect faults in an application and all its dependent components, including the associated database, operating system, network, and storage resources. When a failure is detected, Cluster Server gracefully shuts down the application, restarts it on an available server, connects it to the appropriate storage device, and resumes normal operations.

Veritas Cluster Server is supported on both Red Hat Enterprise Linux and SUSE Linux Enterprise Server. For supported storage, OS versions, and recommended patch levels please see the Hardware Compatibility List or the Veritas Installation Assessment Service at http://vias.symantec.com which can assist with installation/upgrade checking utilities.

Veritas Cluster Server Architecture Introduction

This introduction is an overview of the basic concepts within Veritas Cluster Server. It is intended to provide enough information that would allow users to determine the requirements to migrate a Serviceguard for Linux cluster to Veritas Cluster Server.

Veritas Cluster Server clustering concepts

Cluster

A single VCS cluster consists of multiple servers or systems, either physical or virtual, connected in various combinations to shared storage devices and network connections. VCS monitors and controls applications running in the cluster, and can restart applications in response to a variety of hardware or software faults.

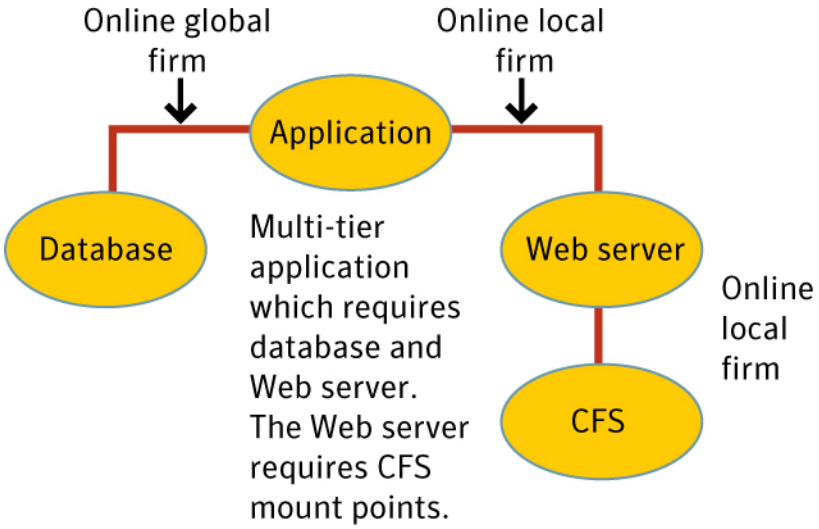
A cluster is defined as all systems that share a common cluster configuration and utilize a common interconnecting network. The VCS cluster interconnect consists of redundant physical Ethernet connections, generally over two or more dedicated private networks. The communications layer carries heartbeats between systems within the cluster, as well as membership and state change information. This will be described in the cluster communications section below.

Applications can be configured to run on specific nodes in the cluster based on priority, application dependencies, or workload policies. Storage is configured to provide access to shared application data for the systems that are hosting the application. In that respect, the actual storage connectivity will determine where applications can be run: Nodes sharing access to storage are “eligible” to run an application. Shared storage is not a requirement for Veritas Cluster Server.

Service Group

A service group is a virtual container that contains all the hardware and software resources that are required to run the managed application. Service groups allow VCS to control all the hardware and software resources of the managed application as a single unit. When a failover occurs, resources do not fail over individually— the entire service group fails over. If there is more than one service group on a system, a group may fail over without affecting the others.

Service groups can be dependent on each other. For example a finance application may be dependent on a database application. Because the managed application consists of all components that are required to provide the service, service group dependencies create more complex managed applications. When using service group dependencies, the managed application is the entire dependency tree. The following is a graphical representation of the Service Group dependencies in a VCS cluster that controls an Application, a Database and a Webserver. The Webserver requires that CFS Mount points are online on the local VCS node before it will come online. The Application requires that the Webserver is running on the local node and that the Database is online somewhere in the cluster before the Application will come online.



In this diagram each item is a Service Group and the lines are the Service Group Dependencies. The service groups are collection of resources, which will be defined further in the document.

Agents

Veritas Cluster Server agents handle the start, stop, and monitoring of all resources contained within a service group. Agents receive instructions regarding what action to take, if any are necessary, from the VCS engine. If any action is necessary, then it will return the results of those actions to the engine. Agents also have the ability to recover from an unknown state. This function within the agent framework is called the clean process.

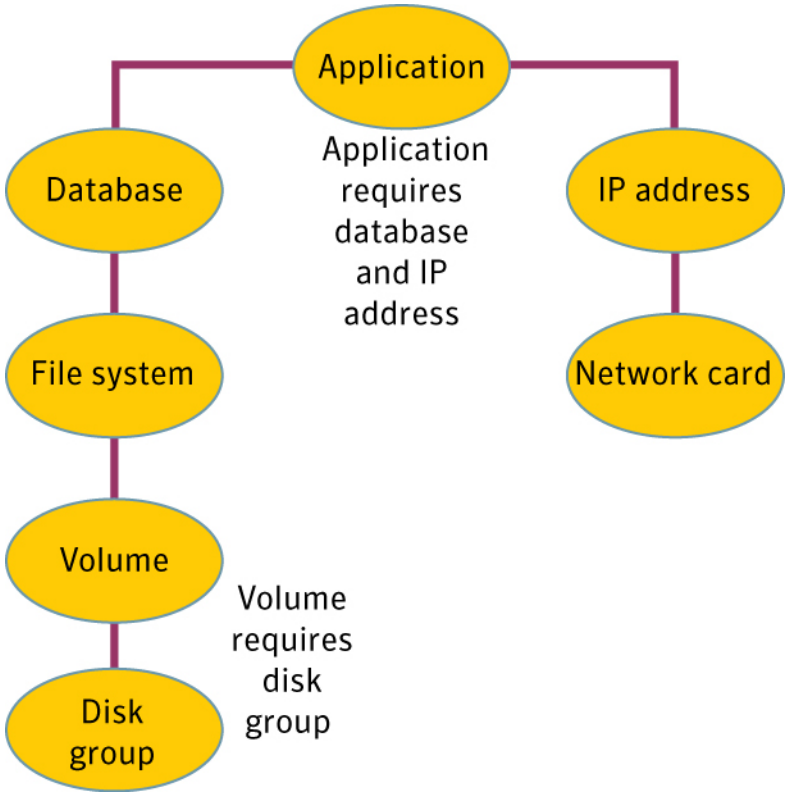
Veritas Cluster Server also ships with agents to control all common system functions, such as file systems and network addresses. Additional agents are provided for out-of-the-box support for most enterprise applications, such as databases, application servers, and Web servers. This includes complete out-of-the-box (no customization required) support for Oracle®, DB2®, Sybase, SAP®, WebSphere, WebLogic, and many other enterprise applications. Please see http://www.symantec.com/business/products/agents_options.jsp?pcid=pcat_business_cont&pvid=20_1 for a complete

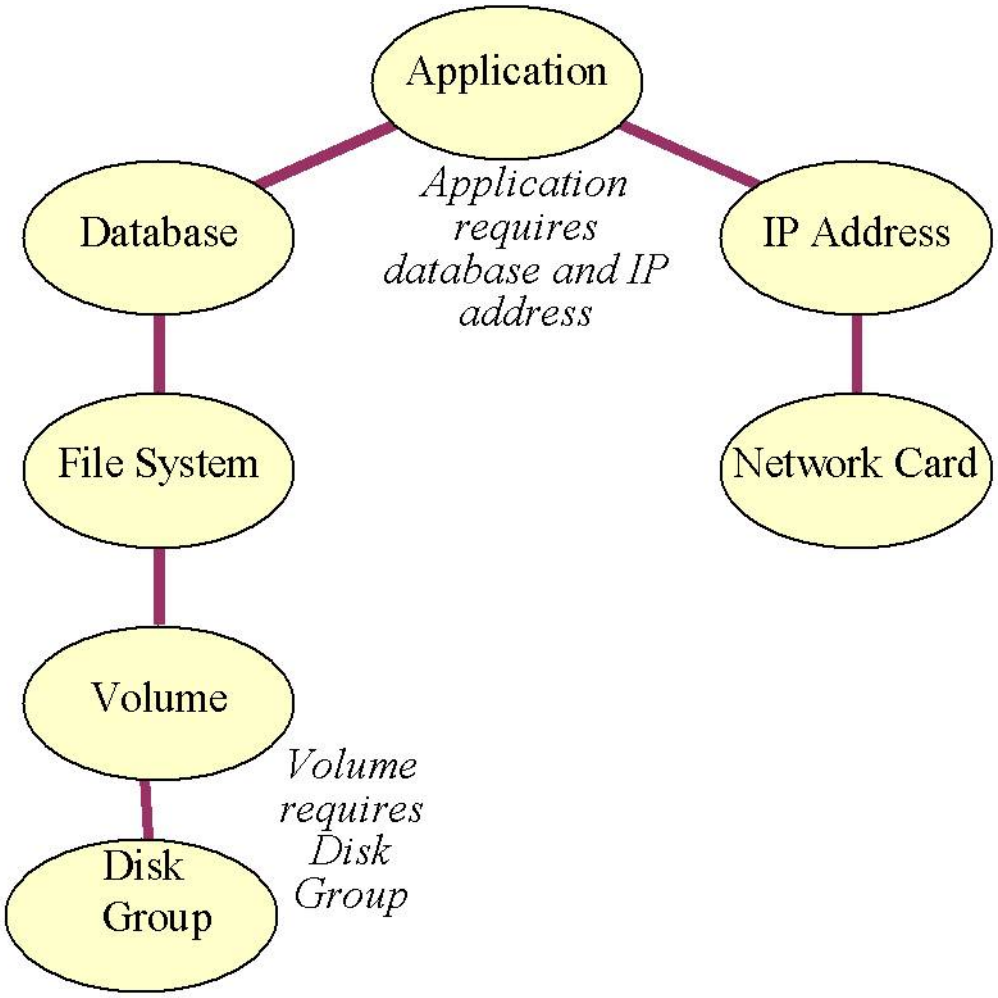
list of applications with existing VCS Agents. All applications that can run in a High Availability environment can utilize the bundled agent that controls applications with their own start and stop scripts. Custom agents can be developed for managing applications with unique and advanced startup, shutdown and monitoring requirements. For more information on agents please see the Veritas Cluster Server Installation Guide.

Resources

Resources are hardware or software entities that make up the application. Types of resources include disk groups and file systems, network interface cards (NIC), IP addresses, and system process. A resource within Veritas Cluster Server is a specific instance of a service controlled by an agent. VCS may control the import of several disk groups and each one is an individual resource.

Each resource has its startup and shutdown order dictated by resource dependencies. This allows for multiple resources to be ordered based on OS or application requirements. For example, a file system resource would need the disk group resource it is contained within to be imported before the file system could be mounted when the service group is starting up. When the VCS Management console or VCS Java GUI is used, a graphical representation of Resource dependencies can be displayed. The following is a graphical example of a service group dependency tree.





Configuration files

Veritas Cluster Server has two primary configuration files located in /etc/VRTSvcs/conf/config.

These two files are main.cf, which is the primary configuration file and types.cf, which is used to define how bundled agents behave. If additional agents are installed and configured, they will have types.cf file specific to their application. For example, if the Oracle agent is in use then a line at the top of the main.cf would include the OracleTypes.cf file to define how the agent is configured:

```
# cat /etc/VRTSvcs/conf/config/main.cf
include "types.cf"
include "OracleTypes.cf"
```

VCS keeps the main.cf and all types.cf files in sync on all nodes in the cluster. The cluster configuration is stored in the previously mentioned files. When the cluster is started those files are validated and read into the HAD process, which will be further discussed in the cluster communication section, on the local cluster node. If the main.cf file is changed while the cluster is online, no changes are introduced in the running cluster. There are two methods to modify the configuration within a running cluster:

1. Run CLI commands to modify the cluster configuration
2. Use the GUI to run commands to modify the cluster configuration

After the cluster is modified and the configuration is closed the changes are written to the main.cf and types.cf files on all nodes in the cluster to ensure all configuration files stay in sync.

Cluster Communication

Veritas Cluster Server uses a cluster interconnect for network communications between cluster systems. Each system runs as an independent unit and shares information at the cluster level. On each system the VCS High Availability Daemon (HAD), which is the decision maker for the cluster, maintains a view of the cluster configuration. This daemon operates as a replicated state machine, which means all systems in the cluster have a synchronized state of the cluster configuration. This is accomplished by the following:

- All systems run an identical copy of HAD.
- HAD on each system maintains the state of its own resources, and sends all cluster information about the local system to all other machines in the cluster.
- HAD on each system receives information from the other cluster systems to update its own view of the cluster.
- Each system follows the same code path for actions on the cluster.

HAD communicates over a high-performance, low-latency replacement for the IP stack consisting of two components, Group Membership Services/Atomic Broadcast (GAB) and Low Latency Transport (LLT). These two components operate in a manner similar to the TCP and IP protocols in that they connect nodes and communicate information between them. In order to make these protocols as efficient as possible, a few layers in the TCP/IP stack have been removed. Because of this GAB and LLT heartbeat traffic is not routable though it can be configured using UDP. The following sections go into more detail on the specific protocols.

Group Membership Services/Atomic Broadcast (GAB)

The Group Membership Services/Atomic Broadcast protocol (GAB) has two major functions.

Cluster membership

- GAB maintains cluster membership by receiving input on the status of the heartbeat from each system via LLT, as described below. When a system no longer receives heartbeats from a cluster peer, LLT passes the heartbeat loss to GAB. GAB marks the peer as DOWN and excludes it from the cluster. In most configurations, membership arbitration is used to prevent network partitions.

Cluster communications

- GAB's second function is reliable cluster communications. GAB provides guaranteed delivery of messages to all cluster systems. The Atomic Broadcast functionality is used by HAD to ensure that all systems within the cluster receive all configuration change messages, or are rolled back to the previous state, much like a database atomic commit. While the communications function in GAB is known as Atomic Broadcast, no actual network broadcast

traffic is generated. An Atomic Broadcast message is a series of point to point unicast messages from the sending system to each receiving system, with a corresponding acknowledgement from each receiving system.

Low Latency Transport (LLT)

The Low Latency Transport protocol has two major functions.

Traffic distribution

- LLT provides the communications backbone for GAB. LLT distributes (load balances) inter-system communication across all configured network links. This distribution ensures all cluster communications are evenly distributed across all network links for performance and fault resilience. If a link fails, traffic is redirected to the remaining links. A maximum of eight network links are supported.

Heartbeat

- LLT is responsible for sending and receiving heartbeat traffic over each configured network link. LLT heartbeat is an Ethernet broadcast packet. This broadcast heartbeat method allows a single packet to notify all other cluster members the sender is functional, as well as provide necessary address information for the receiver to send unicast traffic back to the sender. The heartbeat is the only broadcast traffic generated by VCS. Each system sends 2 heartbeat packets per second per interface. All other cluster communications, including all status and configuration traffic is point to point unicast. This heartbeat is used by the Group Membership Services to determine cluster membership.

Data Protection

Membership arbitration by itself is inadequate for complete data protection because it assumes that all systems will either participate in the arbitration or are already down. Rare situations can arise which must also be protected against. Although implementation of I/O Fencing is optional, it is recommended to protect against potential data corruption. Some examples of mitigated issues are

- A system hang causes the kernel to stop processing for a period of time.
- The system resources were so busy that the heartbeat signal was not sent.
- A break and resume function is supported by the hardware and executed. Dropping the system to a system controller level with a break command can result in the heartbeat signal timeout.

In these types of situations, the systems are not actually down, and may return to the cluster after cluster membership has been recalculated. This could result in data corruption as a system could potentially write to disk before it determines it should no longer be in the cluster.

Combining membership arbitration with data protection of the shared storage eliminates all of the above possibilities for data corruption.

Data protection fences off or removes access to the shared data storage from any system that is not a current and verified member of the cluster. Access is blocked by the use of SCSI-3 persistent reservations.

SCSI-3 Persistent Reservation

SCSI-3 Persistent Reservation (SCSI-3 PR) supports device access from multiple systems, or from multiple paths from a single system. At the same time it blocks access to the device from other systems, or other paths.

Veritas Cluster Server logic determines when to online a service group on a particular system. If the service group contains a disk group, the disk group is imported as part of the service group being brought online. When using SCSI-3 PR, importing the disk group puts registration and reservation on the data disks. Only a system that has imported the storage with SCSI-3 reservation can write to the shared storage. This prevents a system that did not participate in membership arbitration from corrupting the shared storage.

SCSI-3 PR ensures persistent reservations across SCSI bus resets. Membership arbitration combined with data protection is termed I/O Fencing. Coordination Point Server (CPS), Introduced in VCS version 5.1, can be used instead of a physical disk for use with I/O Fencing. CPS takes the place of a single disk. Multiple CPS servers could be used to replace all SCSI-3 PR disks within a cluster. The primary use case for Coordination Point Servers is within a distributed computing environment as the communication occurs over IP.

Note: Use of SCSI 3 PR protects against all components in the IT environment that might be trying to write illegally to storage, not only VCS related elements.

Term Comparison - SGLX and VCS

Term	HP Serviceguard on Linux	VCS on Linux
Cluster	Cluster	Cluster
Cluster Member	Node	System
Framework to used to online, offline and monitor applications controlled by the cluster	Toolkit	Agent
A grouping of application services together	Package	Service Group
A grouping of application services that run on all nodes in the cluster at the same time	Multi-node package or System multi-node package	Parallel Service Group
Heartbeat technologies	Ethernet or Inifiniband	Ethernet
Cluster Split-Brain Protection	Lock LUN or Quorum Server	SCSI-3 PR/Coordination Point Server

Command Comparison - SGLX and VCS

Command Purpose	Serviceguard on Linux	Veritas Cluster Server on Linux
Cluster startup	cmruncl	hastart
Cluster shutdown	cmhaltcl	hastop
Bring Online an application package/group	cmrunpkg	hagrp –online <SG> -sys <System>
Bring Offline an application package/group	cmhaltpkg	hagrp –offline <SG> -sys <System>
Display the cluster status	cmviewcl	hastatus -sum
Additional Terms		SG = Service group System = Cluster Node

Veritas Cluster Server cluster heartbeats

Because VCS communicates using LLT and GAB protocols, it does not use IP communication in the default configuration. This requires that the connections between nodes not be routed and that each heartbeat NIC use a different VLAN. At least 2 NICs are required per cluster for heartbeats. Configurations requiring IP communication (e.g. stretched clusters utilizing WAN links) can alternatively use "LLT over UDP" (see appendix section of the Veritas Cluster Server Install Guide).

Migration Procedure

Planning Phase

In order to ensure a successful transition from SGLX to VCS on Linux several items need to be considered. To begin with the cluster heartbeats and data protection strategies need to be mapped out to determine if the current SGLX heartbeats can be used for VCS. After the cluster communication is documented then each service under Serviceguard control needs to be considered. If a VCS Agent is available for the resource to be controlled, then the appropriate attributes need to be identified to properly control that resource. Each Agent has different attributes used to control resources. For example, an IP resource would require attributes like the NIC card to be used and the NetMask used to configure the IP.

Planning is required to ensure an optimal implementation. The VCS configuration can be generated prior to the migration using the VCS Simulator. This will verify that the VCS configuration is valid and to make certain that all Single Points of Failure (SPOF) are identified and all SGLX services are migrated.

The Planning phase of this document is intended to present a methodology to be used to properly prepare the user to migrate from SGLX to VCS on Linux. Included is a sample migration which will show the steps taken during this process. Please use appropriate care when planning your migration

Identify applications and resources under Serviceguard for Linux control

Identify all resources currently being controlled by the SGLX cluster. These resources are everything from the NIC and failover IP address, to the Volume Group and File Systems, as well as the applications. To properly identify resources for migration, attention is required to understand the available agents using VCS on Linux. The following is a list of Agents available for VCS on Linux based on agent categories:

Application	apache_agent	sapwebas_agent	powercentersvcmgr_agent
	tuxedo_agent	oracleapps_agent	weblogic_agent
	oracleas_agent	websphere_agent	saplivecache_agent
	webspheremq_agent	sapnw_agent	
Database	db2_agent	sapmaxdb_agent	informix_agent
	sybase_agent	oracle_agent	
Replication	dataguard_agent	ntap_agent	db2hadr_agent

	srdf_agent	htc_agent	srdfstar_agent
	metro_mirror_agent	mirrorview_agent	svccopyservices_agent
Storage*	DiskGroup	DiskReservation	Volume
	Mount	LVMlogicalvolume	LVMvolumegroup
Network*	**IP**	**NIC**	**IPMultiNIC**
	MultiNICA	**DNS**	
File Share*	NFS	NFSRestart	Share
	SambaServer	NetBIOS	SambaShare
Service*	**Application**	**Process**	**ProcessOnOnly**
Infrastructure*	NotifierMngr	VRTSWebApp	Proxy
	Phantom	RemoteGroup	
Testing*	**ElifNone**	**FileNone**	**FileOnOff**
	FileOnOnly		
*Agents that are bundled with the Product. All other agents are bundled in a free Agent Pack to allow for updates on a continuous basis			

To determine how to properly implement and the capabilities of each agent please see the Veritas Cluster Server Bundled Agents Reference Guide and the Veritas Cluster Server Agent Pack.

Veritas Cluster Server Hardware Prerequisites

The primary hardware requirement for Veritas Cluster Server is related to cluster communication over heartbeats. VCS requires a minimum of 2 NICs to be used for heartbeats. VCS, as with any installed application, has disk space requirements for each node (server) in the cluster. The current requirements can be found in the VCS Installation Guide for Linux or through the Installation Assessment website – http://vias.symantec.com/

There may be additional hardware requirements to protect the applications and avoid Single Points of Failure (SPOF). When architecting the environment, it is essential to validate that all required resources are examined to guard against error conditions. If SCSI-3 is planned to be included in the environment then the shared storage array will need to have

that feature enabled and disks will need to be assigned to the coordinator disk group. To validate the implementation availability of this feature, please see the Veritas Cluster Server Installation guide for more information on I/O fencing.

Veritas Cluster Server Linux OS Prerequisites

As of January 2010 the following OS versions are supported with version 5.1 of Veritas Cluster Server:

- Red Hat Enterprise Linux 5 (RHEL 5) with Update 3 (2.6.18-128.el5 kernel) or later on AMD Opteron or Intel Xeon EM64T (x86_64)
- SUSE Linux Enterprise Server 10 (SLES 10) with SP2 (2.6.16.60-0.21 kernel) on AMD Opteron or Intel Xeon EM64T (x86_64)
- SUSELinux Enterprise Server 11 (SLES 11) (2.6.27.19-5 kernel) onAMDOpteron or Intel Xeon EM64T (x86_64)
- Oracle Enterprise Linux (OEL 5) with Update 3 (2.6.18-128.el5 kernel) or later on AMD Opteron or Intel Xeon EM64T (x86_64)

Please see the VCS Release notes for the latest details: http://sfdoccentral.symantec.com/index.html

Preparing for Veritas Cluster Server Installation

There are several steps that need to be performed as pre-installation tasks. These include establishing the heartbeat connections, validating shared storage is in place, deciding if SCSI-3 PR for I/O Fencing will be implemented and obtaining a license key depending on the version of VCS to be installed(permanent, temporary or keyless) to be used during installation. For all pre-installation tasks please see the Veritas Cluster Server Installation Guide for Linux.

Implementation Phase

Shutdown the SGLX cluster

Before stopping the cluster, ensure that all packages are offline by running the command:

```
# cmviewcl -v
```

For each package that is still running, issue the command for them to shutdown:

```
# cmhaltpkg <Package_Name>
```

The command cmhaltcl is then used to halt the entire cluster. This command will halt the Serviceguard daemons on all nodes in the cluster and can be issued from any cluster node. You can use the -f option to force the cluster to halt even when packages are running, as in this example:

```
# cmhaltcl -f -v
```

Uninstall SGLX software

To uninstall Serviceguard, run rpm -e on all the rpms you have installed. The uninstall process can be done at a later date to allow for a migration backout plan. VCS and SGLX can be installed on the same box as long as when VCS will control the

applications Serviceguard daemons are disabled, the startup of Serviceguard processes are disabled and only VCS is controlling the application resources. To uninstall the Serviceguard rpm, here is an example:

rpm -e serviceguard-A.11.19.00-0.rhel5.x86-64.rpm

Veritas Cluster Server Cluster Configuration

If I/O Fencing is to be utilized within the VCS cluster then the disks to be used need to be validated, initialized, setup in a disk group and made ready to be included within the configuration. As a note, I/O Fencing requires VxVM. For full instructions on how to setup and validate if SCSI-3 can be used for VCS in the environment, please see the VCS Installation Guide for Linux for further information. VCS can be configured using LVM as the volume manager as well as VxVM.

Veritas Cluster Server installation

Veritas Cluster Server is installed via the Veritas Common Product Installer or installvcs script. For details for usage, please reference the VCS installation procedures as outlined in the Veritas Cluster Server Installation Guide for Linux.

Veritas Cluster Server Heartbeats

Veritas Cluster Server Heartbeats will be established during the binary installation process. The installer script asks which NIC will be used for heartbeats. The NICs can be different on each node in the cluster but it is preferred to have the configurations be as similar as possible. VCS Heartbeats need to be on separate networks or VLAN to add redundancy and reduce the possibility of a single LAN causing all Heartbeat links to go down at once.

Veritas Cluster Server Cluster Creation

Veritas Cluster Server can be modified using 3 different methods: Java Graphical User Interface (GUI), a connection to the VCS Management Console or VCS using the Command Line Interface (CLI). All three can modify an already running cluster or to edit the cluster configuration file (main.cf) the cluster needs to be offline.

During VCS installation a configuration file is created. It contains the systems that were designated during the installation process. It may also contain services to send out SNMP/SMTP alerts if they were configured during installation.

It is at this point that the information from the SGLX cluster needs to be migrated into the VCS cluster. Each service in Serviceguard, which is necessary for the application to function will need to be established within VCS. These services can be implemented using any of the three methods to modify the VCS cluster. Examples of this are provided in a later section of this document.

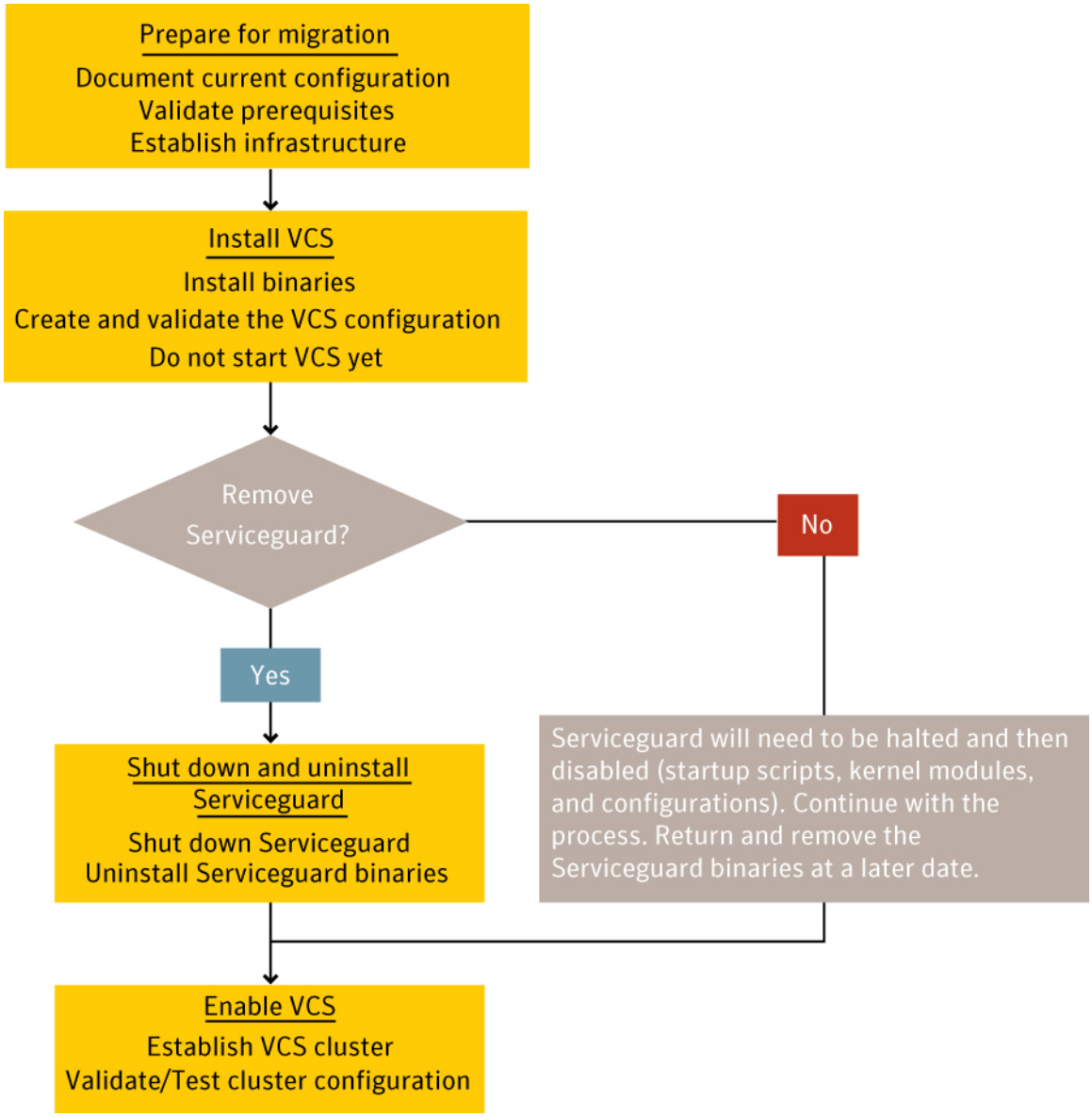
Verify the Veritas Cluster Server Cluster

When all of the services migrated from Serviceguard are now configured within VCS, several additional steps should be taken to ensure the ability to properly administer the cluster. These steps include adding VCS users with appropriate privileges and determining which method will be used to control the cluster.

Veritas Cluster Server validation and testing

A plan needs to be established to validate the cluster functionality. VCS has local HA Fire Drill capabilities that can be used to determine if the cluster was setup properly. In addition to using the Fire Drill function, cluster testing should be performed to confirm that the configuration acts as expected.

Serviceguard for Linux to VCS Flowchart



Methods of Controlling the Veritas Cluster Server Cluster

There are several methods of managing the Veritas Cluster Server Clusters. Historically VCS administrators can use the command line to control the cluster, maintain the configuration and monitor the status of the cluster. This option of management continues to be available. There are two major disadvantages of the CLI, knowledge of the commands to be used and their lack of graphical representation. Since VCS was introduced it has also included a GUI that could be

installed to manage the cluster in a graphical mode. The Java console enables the cluster to be managed by users from their local PCs.

Along with the command line for VCS and the Java Console, Symantec has developed management utilities that enable features beyond just controlling the cluster. VCS Management Console allows for the management of several clusters at once. It supports advanced capabilities in reporting and configuration checking utilities. Storage Foundation Manager is an additional console that includes the ability to maintain Storage Foundation as well as VCS with limited functionality in one application. The final Console in the Symantec management strategy is the Veritas Operations Manager (available 2Q2010), which is the combination of Storage Foundation Manager and VCS Management Console. VOM will also include capabilities not found within SFM or VCS MC such as the ability to determine appropriate patches, run reports on VCS trends and detect barriers to successful failover, both for Global clusters as well as local clusters.

With all of the management utilities available to maintain the VCS cluster, they are included with Veritas Cluster Server.

Summary of VCS management capabilities

VCS Command Line Interface (CLI)

- Single cluster management UI
- Commands are consistent across Operating Systems and VCS versions
- Every node in the cluster can be used to run commands against the cluster
- No additional packages are required for use

VCS Java Console

- Single cluster management UI
- Will no longer be packaged with SFHA/VCS when version 5.1 is GA and will be available as a download from http://go.symantec.com/vcsmc

VCS Cluster Simulator

- Veritas Cluster Server Simulator helps administrators simulate high availability environments from their laptops
- It enables the ability to test multiple application failover scenarios without impacting production
- Creating cluster configurations simplify installations as the configuration is available to test before installation.
- The download location for the cluster simulator also contains a flash demo on the product

VCS Management Console

- Multi-Cluster Management and Reporting tool
- Supports stretch clusters and global clusters including site-to-site migration and DR
- Includes proactive checks with Firedrill scheduling
- Management Server installs on Win, Linux and Solaris

- Can be downloaded from http://go.symantec.com/vcsmc
- More on SCORE: http://score.corp.symantec.com/products/289

Storage Foundation Manager

- Provides single pane of glass views from App to Cluster to Server and Storage infrastructure for all SFHA
- Provides monitoring, reporting and operations for SF
- Can be downloaded from http://go.symantec.com/vom
- More on SCORE: http://score.corp.symantec.com/products/237
- Full VCS support will come through the VOM roadmap

Veritas Operations Manager

- Complete management solution for SF/HA environments
- Includes visibility, monitoring and reporting for SF and VCS
- Cross-stack fault and risk detection capabilities
- Connection to VOS for available patches
- Is built on top of SFM architecture and will be released in First Half of 2010

Appendix reference information

Migration Planning – VCS Cluster Information

This section is provided as a sample cluster configuration form. Information gathered from the Serviceguard on Linux cluster can be used to configure Veritas Cluster Server. Fill in as much data as possible in the below forms to ease the VCS cluster configuration. For more information on how to implement VCS please see the earlier portions of this document or the Veritas Cluster Server for Linux Installation guide.

LVM Volume Group Information

Volume Group Name: _____

Logical Volume Name: _____ Mount Point: _____

Logical Volume Name: _____ Mount Point: _____

Logical Volume Name: _____ Mount Point: _____

Volume Group Name: _____

Logical Volume Name: _____ Mount Point: _____

Logical Volume Name: _____ Mount Point: _____

Logical Volume Name: _____ Mount Point: _____

Volume Group Name: _____

Logical Volume Name: _____ Mount Point: _____

Logical Volume Name: _____ Mount Point: _____

Logical Volume Name: _____ Mount Point: _____

VxVM Disk Group

Disk Group Name: _____

Volume Names: _____ Mount Point: _____

Volume Names: _____ Mount Point: _____

Volume Names: _____ Mount Point: _____

Disk Group Name: _____

Volume Names: _____ Mount Point: _____

Volume Names: _____ Mount Point: _____

Volume Names: _____ Mount Point: _____

Disk Group Name: _____

Volume Names: _____ Mount Point: _____

Volume Names: _____ Mount Point: _____

Volume Names: _____ Mount Point: _____

VCS Cluster Information

Cluster Name: _____

Cluster Nodes: _____

Cluster Number: _____

CVM/Oracle RAC Information

Oracle RAC Version: _____ Node Names running RAC: _____

CVM Disk Group(s): _____

CVM Volume(s): _____

Heartbeat Information

Heartbeat NICs (2 minimum): _____ _____ _____ _____ _____

SCSI-3 I/O Fencing Information (SCSI-3 is Optional)

SCSI-3 Disk Group: _____

SCSI-3 Disks (3 minimum): _____ _____ _____

_____ _____ _____ _____

Have these Devices been tested with the vxfentsthdw command?(Yes/No) _____

GUI User Security (User Admin is created by defult

User name: _____

Access: _____ Cluster Admin/Cluster Operator/Group Admin/Group Operator/Guest

Service group: _____ required if Group Admin or Group Operator

Service group Information – Needed for each package configured within Serviceguard

Service group Name: _____ Service group type (Parallel or Failover): _____

System List: _____

Auto Start List: _____

Resources:

Resource Name: _____ Type of Resource/Agent to be used: _____

Resource Attributes (Different for each type of resource): _____

_____

Is the Resource Critical? _____(If a resource is marked critical, when it faults, VCS will failover the servicegroup to another node in the cluster.)

Resource Name: _____ Type of Resource/Agent to be used: _____

Resource Attributes (Different for each type of resource): _____

_____

Is the Resource Critical? _____

Resource Name: _____ Type of Resource/Agent to be used: _____

Resource Attributes (Different for each type of resource): _____

_____

Is the Resource Critical? _____

Resource Name: _____ Type of Resource/Agent to be used: _____

Resource Attributes (Different for each type of resource): _____

_____

Is the Resource Critical? _____

Resource Name: _____ Type of Resource/Agent to be used: _____

Resource Attributes (Different for each type of resource): _____

_____

Is the Resource Critical? _____

Dependencies (What is the order of startup)

Resource named: _____ requires that _____ resource is online first

Resource named: _____ requires that _____ resource is online first

Resource named: _____ requires that _____ resource is online first

Resource named: _____ requires that _____ resource is online first

Resource named: _____ requires that _____ resource is online first

Service group Dependencies (What is the dependency between service groups?)

Service group named: _____ requires that _____ service group is

_____ (online or offline) _____ (local or global) _____ (firm – if this mandatory)

Network Information (goes in the appropriate service groups):

Virtual IP _____ subnet mask_____ associated NIC: _____

Network Hosts: _____ a List of IPs Used to tests if a NIC is online by ping

Notification Information:

SMTP Server: _____

SMTP Recipients: _____ Notification Level: _____-Information/Warning/Error

SMTP Recipients: _____ Notification Level: _____-Information/Warning/Error

SMTP Recipients: _____ Notification Level: _____-Information/Warning/Error

SMTP Recipients: _____ Notification Level: _____-Information/Warning/Error

SNMP Server: _____

Attribute Information

Each level within the cluster has default values. These attributes can be modified to enable the preferred behavior. The following is a sample of attributes that can be modified. For a full listing please see the Veritas Cluster Server Administrators Guide.

Agent Attributes:

An Agent is the binary that controls an application or process. This control of an application is the startup, shutdown, monitor and clean procedures. Each Agent has specific attributes necessary to control and monitor the application/ process defined. When there is a specific instance of an application, for example a NIC card, then that is a resource. There are additional attributes that are used with the agent to control how it functions. The following are a couple of default variables that can be modified to control how the cluster behaves on a per Agent basis:

MonitorInterval (How often is a resource monitored?) _____ Default 60 (seconds)

OfflineMonitorInterval (Same as MonitorInterval but on the Offline node) _____ Default 300 (seconds)

RestartLimit (The number of times a resource can restart before failing) ____ Default 0

OnlineRetryLimit (The limit in attempting to bring a resource online during startup) ____ Default 0

Resource Attributes:

Each Resource has the attributes to control an application using an Agent. For example a Mount Resource requires information on the specific File System to be managed. Beyond the specific information passed to the Agent to manage the Resource there are default values that change the behavior of service group. Here is an example of an attributes that can be modified for each Resource:

Critical (This specifies if the resource goes offline unexpectedly it will cause the service group to failover)

Step-by-step migration with sample applications – SGLX -> VCS

Migration Steps

1. Perform pre-planning steps to gather existing configuration information and application information to migrate to Veritas Cluster Server

- This includes ensuring that VCS installation binaries and a license key are available unless keyless licensing is to be used with VCS 5.1.
- There are additional pre-planning steps needed to utilize certain features within the product such as the cluster management console and authentication broker. For additional information please see the Veritas Cluster Server on Linux installation guide.
- To install all nodes within a cluster at one time, trusted SSH communication needs to be in place before VCS is installed.

2. Validate Heartbeat network communication

- Ensure that NICs can communicate only to their corresponding pair. The communication should not be possible across NICs, for example Heartbeat NIC1 on Node1 should only be able to communicate to HeartbeatNIC1 on Node 2 and not be able to communicate to HeartbeatNIC2 on Node2.

3. Bring the Serviceguard cluster down on all nodes and disable the cluster from startup

- Run the command: cmhaltcl
- Backup all Serviceguard configuration files
- Move all of the startup and shutdown commands for Serviceguard out of place. These are typically located in /etc/rc2.d and need to be moved so they do not bring up Serviceguard upon a reboot.

4. Install the VCS software

- With the CD in place run installer or go into the cluster_server directory and run installvcs
- Continue through the installation menus with information regarding the cluster setup gathered in the pre-planning steps. You have a choice when using the installer to just install the binaries (RPMs) or to install the binaries and configure the cluster. If the pre-planning phase has been completed, the install and configure option should be selected. The info required for use with this installation and configuration method includes:
    - License Key (unless Keyless Licensing is to be used with VCS version 5.1)
    - Cluster name and number to be used (The name and id must be unique.)
    - Heartbeat NICs
    - Will the Symantec Product Authentication Service be used? If not then the configuration of VCS Users (Username, User Access, Password)
    - Establishing communication with a Cluster Management Console if one is to be used
    - The Setup of SNMP and SMTP notification if these will be used

5. At this point the cluster has been established and a base configuration was created. Our next step is to configure the Services under SGLX within VCS. As a note, this step can be done prior to VCS binaries being installed to reduce downtime

- Take each Serviceguard Service and port it to VCS
- Examples of this can be seen in the Appendix that shows the output of the SGLX configuration files and the VCS Configuration files
- Depending on your environment you can edit the configuration files manually or when the cluster is active, use the CLI, Java GUI or the Cluster Server Console to configure the cluster.

6. With any cluster software installation validate that it is configured correctly.

7. In our example we moved the startup scripts out of place in Step #3 rather than uninstalling the Serviceguard binaries. When the migration is complete and tested Serviceguard needs to be uninstalled. Put the original files back in place and uninstall the Serviceguard packages.

SGLX Configuration Files Examples

The following files were used in setting up a generic configuration within SGLX used to cluster an NFS share. We have a two node cluster (redhat1 and redhat2) that manages a single package (nfs-pkg). The comments have been removed to keep the example brief.

pkg-nfs.conf (for brevity all comments as well as executable lines of code are not included):

```
PACKAGE_NAME         nfs1
PACKAGE_TYPE         FAILOVER

NODE_NAME         redhat1
NODE_NAME         redhat2

AUTO_RUN         YES
NODE_FAIL_FAST_ENABLED     NO

RUN_SCRIPT         /usr/local/cmcluster/nfs1/pkg-nfs.cntl
HALT_SCRIPT         /usr/local/cmcluster/nfs1/pkg-nfs.cntl

RUN_SCRIPT_TIMEOUT              NO_TIMEOUT
HALT_SCRIPT_TIMEOUT             NO_TIMEOUT
SUCCESSOR_HALT_TIMEOUT     NO_TIMEOUT

FAILOVER_POLICY         CONFIGURED_NODE
FAILBACK_POLICY         MANUAL
PRIORITY                      NO_PRIORITY

MONITORED_SUBNET         192.168.1.0
MONITORED_SUBNET_ACCESS         FULL

SERVICE_NAME                nfs1.monitor
```

pkg-nfs.cntl (including set variables):

```
VG[0]="nfsvg"
LV[0]="/dev/nfsvg/nfslv"; FS[0]="/nfs"; FS_TYPE[0]="ext3"; FS_MOUNT_OPT[0]="-o rw"

IP[0]="192.168.1.5"
SUBNET[0]="192.168.1.0"
```

pkg-nfs.cntl (for brevity all comments as well as executable lines of code are not included):

```
GFS="NO"DATA_REP="none"
VGCHANGE="vgchange -a y"                # Default
VG[0]="nfsvg"
LV[0]="/dev/nfsvg/nfslv"; FS[0]="/nfs"; FS_TYPE[0]="ext3"; FS_MOUNT_OPT[0]="-o rw"
FS_UMOUNT_OPT[0]=""; FS_FSCK_OPT[0]=""
```

```
FS_UMOUNT_COUNT=1
FS_MOUNT_RETRY_COUNT=0
IP[0]="192.168.1.5"
SUBNET[0]="192.168.1.0"
HA_APP_SERVER="pre-IP"
# START OF CUSTOMER DEFINED FUNCTIONS
# END OF CUSTOMER DEFINED FUNCTIONS
```

Veritas Cluster Server Configuration Files Examples

The following are the files used in the configuration of Veritas Cluster Server for the same NFS package; within VCS it is called a service group.

/etc/llttab:

```
set-node /etc/VRTSvcs/conf/sysname
set-cluster 200
link eth1 eth1 - ether - -
link eth2 eth2 - ether - -
```

/etc/VRTSvcs/conf/sysname:

redhat1

/etc/llthosts:

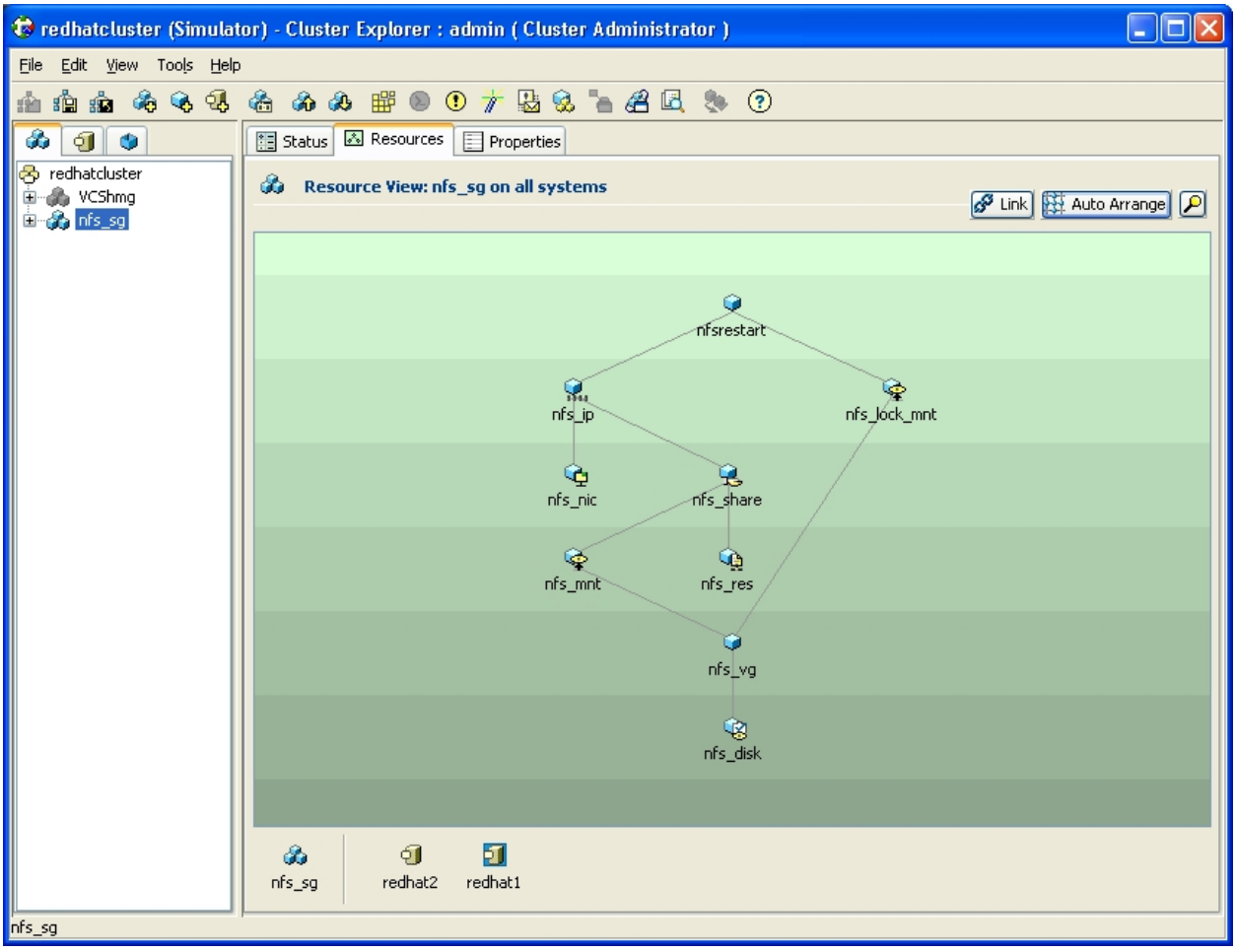
```
0 redhat1
1 redhat2
```

```
include "types.cf"

cluster redhatcluster (
 UserNames = { admin = HopHojOlpKppNxpJom }
 Administrators = { admin }
 )

system redhat1 (
 )

system redhat2 (
 )

group nfs_sg (
 SystemList = { redhat1 = 0, redhat2 = 1 }
 AutoStartList = { redhat1 }
 )
```

```
DiskReservation nfs_disk (
 Disks = { "/dev/sdd" }
 )

IP nfs_ip (
 Device = eth0
 Address = "192.168.1.5"
 NetMask = "255.255.240.0"
 )

LVMVolumeGroup nfs_vg (
 VolumeGroup = nfsvg
 StartVolumes = 1
 )

Mount nfs_mnt (
 MountPoint = "/nfs"
 BlockDevice = "/dev/nfsvg/nfsvol"
 FSType = ext3
 MountOpt = rw
 FsckOpt = "-y"
 )

Mount nfs_lock_mnt (
 MountPoint = "/NFS_lockinfo"
 BlockDevice = "/dev/nfsvg/nfs_lock_vol"
 FSType = ext3
 MountOpt = rw
 FsckOpt = "-y"
 )

NFS nfs_res (
 Address = "192.168.1.5"
 )

NFSRestart nfsrestart (
 NFSRes = nfs_res
 LocksPathName = "/NFS_lockinfo"
 NFSLockFailover = 1
 )

NIC nfs_nic (
 Device = eth0
 )
```

```
Share nfs_share (
 PathName = "/nfs"
 Options = "-o rw"
 )

nfs_ip requires nfs_share
nfs_ip requires nfs_nic
nfs_vg requires nfs_disk
nfs_mnt requires nfs_vg
nfs_lock_mnt requires nfs_vg
nfsrestart requires nfs_lock_mnt
nfsrestart requires nfs_ip
nfs_share requires nfs_mnt
nfs_share requires nfs_res


// resource dependency tree
//
// group nfs_sg
// {
// NFSRestart nfsrestart
//    {
//    Mount nfs_lock_mnt
//       {
//       LVMVolumeGroup nfs_vg
//          {
//          DiskReservation nfs_disk
//          }
//       }
//    IP nfs_ip
//       {
//       Share nfs_share
//          {
//          Mount nfs_mnt
//             {
//             LVMVolumeGroup nfs_vg
//                {
//                DiskReservation nfs_disk
//                }
//             }
//          NFS nfs_res
```

```
//         }
//       NIC nfs_nic
//       }
//   }
//
```



The above diagram is a graphical representation of the resources within a service group and their dependencies. This view is generated from the VCS Java GUI.

Serviceguard and Veritas Cluster Server Configuration Files Migration Example

	Serviceguard configuration pkg-nfs.conf	Veritas Cluster Server configuration main.cf
Name of Package for Serviceguard. Name of Service Group for VCS.	# "PACKAGE_TYPE" is the type of package. # # The PACKAGE_TYPE attribute specifies the desired behavior for this # package. Legal values and their meaning are described below: # # FAILOVER package runs on one node at a time and if a failure # occurs it can switch to an alternate node. # # MULTI_NODE package runs on multiple nodes at the same time and # can be independently started and halted on # individual nodes. Failures of package components such # as services, EMS resources or subnets, will cause # the package to be halted only on the node on which the # failure occurred. Relocatable IP addresses cannot be # assigned to "multi_node" packages. # # SYSTEM_MULTI_NODE # package runs on all cluster nodes at the same time. # It cannot be started and halted on individual nodes. # Both "NODE_FAIL_FAST_ENABLED" and "AUTO_RUN" # must be set to "YES" for this type of package. All # "SERVICES" must have "SERVICE_FAIL_FAST_ENABLED" set # to "YES". SYSTEM_MULTI_NODE packages are only # supported for use by applications provided by # Hewlett-Packard. PACKAGE_TYPE FAILOVER	group nfs_sg (SystemList = { redhat1 = 0, redhat2 = 1 } AutoStartList = { redhat1 })
Node Name for Serviceguard. SystemList for VCS.	# "NODE_NAME" specified which nodes this package can run on. # # Enter the names of the nodes configured to run this package, repeat # this line for each cluster member node configured to run this package. # # NOTE: The order in which the nodes are specified here determines the # order of priority when Serviceguard is deciding where to run the # package. # # Example : NODE_NAME first_priority_node # NODE_NAME second_priority_node # # If all nodes in the cluster can run the package, and order is not # important, specify "NODE_NAME *". # # Example : NODE_NAME * # # Legal values for NODE_NAME: # "*", or any node name in the cluster. # Node name is any string that starts and ends with an alphanumeric # character, and contains only alphanumeric characters, dot(.), dash(-), # or underscore(_) in between. # Maximum name length is 39 characters. # NODE_NAME redhat1 NODE_NAME redhat2	Configured in the above stanza. It is defined as an attribute of the service group definition.
Auto Run on Serviceguard AutoStart on VCS	# "AUTO_RUN" defines whether the package is to be started when the # cluster is started, and if it will fail over automatically. # # Possible values are "YES" and "NO". # The default for "AUTO_RUN" is "YES", meaning that the package will be # automatically started when the cluster is started, and that, in the # event of a failure the package will be started on an adoptive node. If # "AUTO_RUN" is "NO", the package is not started when the cluster # is started, and must be started with the cmrunpkg command. # # "AUTO_RUN" replaces "PKG_SWITCHING_ENABLED". # # Legal values for AUTO_RUN: YES, NO. AUTO_RUN YES	AutoStart is enabled by default because it is the default setting it is not listed in the main.cf file. In order for a service group to be started automatically, at least one system will need to be listed in the AutoStartList, which is a variable for the service group.
Fast Failover on Serviceguard There is no Equivelant	# "NODE_FAIL_FAST_ENABLED" will cause node to fail if package fails. # # Possible values are "YES" and "NO". # The default for "NODE_FAIL_FAST_ENABLED" is "NO". In the event of # failure, if "NODE_FAIL_FAST_ENABLED" is set to "YES", Serviceguard	None

within VCS	# will halt the node on which the package is running. All # "SYSTEM_MULTI_NODE" packages must have "NODE_FAIL_FAST_ENABLED" set to # "YES". # # # Legal values for NODE_FAIL_FAST_ENABLED: YES, NO. NODE_FAIL_FAST_ENABLED NO	
Run and Halt Scripts in Serviceguard that startup the resources VCS Agents.	# "RUN_SCRIPT" is the script that starts a package. # "HALT_SCRIPT" is the script that stops a package. # # Enter the complete path for the run and halt scripts. The scripts must # be located in directory with "cmcluster" in the path name. In most cases # the run script and halt script specified here will be the same script, # the package control script generated by the cmmakepkg command. This # control script handles the run(ning) and halt(ing) of the package. # # Legal values for RUN_SCRIPT: # Full path name for the run script with "cmcluster" in the path name. # The maximum length for the path name is MAXPATHLEN characters long. # RUN_SCRIPT /usr/local/cmcluster/nfs1/pkg-nfs.cntl # Legal values for HALT_SCRIPT: # Full path name for the halt script with "cmcluster" in the path name. # The maximum length for path name MAXPATHLEN characters long. # HALT_SCRIPT /usr/local/cmcluster/nfs1/pkg-nfs.cntl	Each resource in VCS has their own Agent binaries to control resource management.
Timeout values in Serviceguard Variables within VCS	# "RUN_SCRIPT_TIMEOUT" is the number of seconds allowed for the package to start. # "HALT_SCRIPT_TIMEOUT" is the number of seconds allowed for the package to halt. # # # If the start or halt function has not completed in the specified # number of seconds, the function will be terminated. The default for # each script timeout is "NO_TIMEOUT". Adjust the timeouts as necessary # to permit full execution of each function. # # Note: The "HALT_SCRIPT_TIMEOUT" should be greater than the sum of # all "SERVICE_HALT_TIMEOUT" values specified for all services. # # Legal values for RUN_SCRIPT_TIMEOUT: NO_TIMEOUT, (value > 0). RUN_SCRIPT_TIMEOUT NO_TIMEOUT # Legal values for HALT_SCRIPT_TIMEOUT: NO_TIMEOUT, (value > 0). HALT_SCRIPT_TIMEOUT NO_TIMEOUT # "SUCCESSOR_HALT_TIMEOUT" limits the amount of time Serviceguard waits # for packages that depend on this package ("successor packages") to # halt, before running the halt script of this package. # # SUCCESSOR_HALT_TIMEOUT limits the amount of time # Serviceguard waits for successors of this package to # halt, before running the halt script of this package. # This is an optional parameter. # Permissible values are 0 - 4294 (specifying the maximum # number of seconds Serviceguard will wait). # The default value is "NO_TIMEOUT", which means Serviceguard # will wait for as long as it takes for the successor package to halt. # The timeout of 0 indicates, that this package will halt without # waiting for successors packages to halt # Example: \n" # SUCCESSOR_HALT_TIMEOUT NO_TIMEOUT # SUCCESSOR_HALT_TIMEOUT 60 # # Legal values for SUCCESSOR_HALT_TIMEOUT: NO_TIMEOUT, ((value >= 0) && (value <= 4294)). SUCCESSOR_HALT_TIMEOUT NO_TIMEOUT	VCS has several variables to control timeouts. These variables are set at different levels of the cluster. There are variables for the Cluster, Each individual Service Group as well as each Resource Type. This allows for the agent that controls the IP resource to have different timeouts compared to an application resource that have different requirements for time to come online and offline. In addition, the default values can be changed for an Agent or Resource Type.
Log files	# "SCRIPT_LOG_FILE" is the full path name for the package control script # log file. The maximum length of the path name is MAXPATHLEN characters long. # # If not set, the script output is sent to a file named by appending # ".log" to the script path. # # Legal values for SCRIPT_LOG_FILE: <Any String> #SCRIPT_LOG_FILE	By default the VCS log files are located at /var/VRTSvcs/log.

Failover Policy	# "FAILOVER_POLICY" is the policy to be applied when package fails. # # This policy will be used to select a node whenever the package needs # to be started or restarted. The default policy is "CONFIGURED_NODE". # This policy means Serviceguard will select nodes in priority order # from the list of "NODE_NAME" entries. # # An alternative policy is "SITE_PREFERRED". This policy means # that when selecting nodes from the list of "NODE_NAME" entries, # Serviceguard will give priority to nodes that belong to the site the # package last ran on, over those that belong to a different site. # # Another policy is "MIN_PACKAGE_NODE". This policy means # Serviceguard will select from the list of "NODE_NAME" entries the # node, which is running fewest packages when this package needs to # start. # # Legal values for FAILOVER_POLICY: CONFIGURED_NODE, MIN_PACKAGE_NODE, SITE_PREFERRED. FAILOVER_POLICY CONFIGURED_NODE	There are several failover policies in VCS. They include Priority, Round Robin and Load. Priority is default and as such is not shown in the standard main.cf file. The order of priority is defined in the SystemList Variable. In our main.cf, that would be: SystemList = { redhat1 = 0, redhat2 = 1 } The system redhat1 is the first priority because it has the lowest value (0).
Failback Policy	# "FAILBACK_POLICY" is the action to take when a package is not running # on its primary node. # # This policy will be used to determine what action to take when a # package is not running on its primary node and its primary node is # capable of running the package. The default policy is "MANUAL". The # "MANUAL" policy means no attempt will be made to move the package back # to its primary node when it is running on an adoptive node. # # The alternative policy is "AUTOMATIC". This policy means Serviceguard # will attempt to move the package back to its primary node as soon as # the primary node is capable of running the package. # # # Legal values for FAILBACK_POLICY: MANUAL, AUTOMATIC. FAILBACK_POLICY MANUAL	In VCS there is no automatic failback option.
Priority of Packages	# "PRIORITY" specifies the PRIORITY of the package. # # This is an optional parameter. Valid values are a number between # 1 and 3000 or NO_PRIORITY. Default is NO_PRIORITY. # A smaller number indicates higher priority. A package with a # numerical priority has higher priority than a package with NO_PRIORITY. # # If a number is specified, it must be unique in the cluster. # To help assign unique priorities, HP recommends you use # priorities in increments of 10. This will allow you # to add new packages without having to reassign priorities. # # Multi-node and System multi node packages cannot be assigned a priority. # # This parameter is used only when a weight has been defined for a package, # a package depends on other packages, # or other packages depend on this package, but can be specified even # when no weights or dependencies have yet been configured. # If priority is not configured, the package is assigned the default # priority value, NO_PRIORITY. # # Serviceguard gives preference to running the higher priority package. # This means that, if necessary, Serviceguard will halt a package (or # halt and restart on anther node) in order to run a higher priority # package. The reason may be: # * the node's capacity would otherwise be exceeded # * there is a direct or indirect dependency between the lower and # higher priority packages. # # For example, suppose package pkg1 depends on package pkg2 # to be up on the same node, both have package switching enabled # and both are currently up on node node1. If pkg1 needs to # fail over to node2, it will also need pkg2 to move to node2. # If pkg1 has higher priority than pkg2, it can force pkg2 to # move to node2. Otherwise, pkg1 cannot fail over because pkg2 is # running on node1. # Examples of package priorities and failover results: # # pkg1 priority pkg2 priority results # 10 20 pkg1 is higher; fails over # 20 10 pkg1 is lower; will not fail over # any number NO_PRIORITY pkg1 is higher; fails over # NO_PRIORITY NO_PRIORITY equal priority; will not fail over # NO_PRIORITY any number pkg1 is lower; will not fail over # # Legal values for PRIORITY: NO_PRIORITY, ((value >= 1) && (value <= 3000)).	VCS does not assign a priority to each Service Group. VCS uses Service Group Dependencies to enforce interconnections between Service Groups.

	PRIORITY NO_PRIORITY	
Dependencies	# The package dependency parameters are "DEPENDENCY_NAME", # "DEPENDENCY_CONDITION" and "DEPENDENCY_LOCATION". # # Dependencies are used to describe the relationship between two packages. # To define a dependency, "DEPENDENCY_NAME" and "DEPENDENCY_CONDITION" # are required and "DEPENDENCY_LOCATION is optional. # # Example 1 : To specify a "same_node" dependency between pkg1 and pkg2: # pkg1's ascii configuration file: # # DEPENDENCY_NAME pkg2_dep # DEPENDENCY_CONDITION pkg2 = up # DEPENDENCY_LOCATION same_node # < Entry concatenated for brevity sake> #DEPENDENCY_NAME #DEPENDENCY_CONDITION #DEPENDENCY_LOCATION	VCS has package dependencies, which are included within the main.cf too. Here is an example of a SAP Service Group requiring another SAP instance to be online on any of the nodes (including the same node). group SAP70I (SystemList = { system1 = 0, system2 = 1 } AutoStartList = { system1 }) <…> **requires group SAP70 online global soft** SAP70 DVEBS02_sap requires SAP70 _cvm_Proxy <…>
Network resource	# "MONITORED_SUBNET" specifies the addresses of subnets that are to be # monitored for this package. # # Enter the network subnet name that is to be monitored for this package. # Repeat this line as necessary for additional subnets. If any of # the subnets defined goes down, the package will be switched to another # node that is configured for this package and has all the defined subnets # available. # # "MONITORED_SUBNET" replaces "SUBNET". # # The MONITORED_SUBNET names can be IPv4 or IPv6, or a mix of both. # # Example : # MONITORED_SUBNET 192.10.25.0 # (netmask=255.255.255.0) # MONITORED_SUBNET 2001::/64 # (netmask=ffff:ffff:ffff:ffff::) # MONITORED_SUBNET 2001:: # (netmask=ffff:ffff:ffff:ffff::) # # Legal values for MONITORED_SUBNET: <Any String> # "MONITORED_SUBNET_ACCESS" defines how the MONITORED_SUBNET is # configured in the cluster. # # # MONITORED_SUBNET_ACCESS defines whether access to a MONITORED_SUBNET # is configured on all of the nodes that can run this package, or only # some. Possible values are "PARTIAL" and "FULL". "PARTIAL" means that # the MONITORED_SUBNET is expected to be configured on one or more of # the nodes this package can run on, but not all. "FULL" means that the # MONITORED_SUBNET is expected to be configured on all the nodes that # this package can run on. "FULL" is the default. (Specifying "FULL" is # equivalent to not specifying the monitored_subnet_access at all.) # # The MONITORED_SUBNET_ACCESS is defined per MONITORED_SUBNET entry. # # Example : # MONITORED_SUBNET 192.10.25.0 # MONITORED_SUBNET_ACCESS PARTIAL # 192.10.25.0 is available on one # # or more nodes of the cluster, # # but not all. # # MONITORED_SUBNET 192.10.26.0 # no MONITORED_SUBNET_ACCESS entry, # # hence this subnet is available # # on all nodes of the cluster. # MONITORED_SUBNET 2001::/64 # MONITORED_SUBNET_ACCESS FULL # 2001::/64 is available on all # # nodes of the cluster. # # Legal values for MONITORED_SUBNET_ACCESS: PARTIAL, FULL. MONITORED_SUBNET 192.168.1.0 MONITORED_SUBNET_ACCESS FULL	NIC nfs_nic (Device = eth0) VCS will monitor the subnet defined on eth0. The default configuration is to ping the broadcast address and if one device responds then mark the resource as online. An optimal configuration utilizes a setting called network hosts. This is a list of hosts on the network used to determine if the NIC is online. Only the listed hostnames or IP addresses will be pinged to determine the state vs. the entire subnet. An example of this configuration would be: NIC nfs_nic (Device = eth0 NetworkHosts={192.168.1.15,192.168.1.30})
Servicename	# "SERVICE_NAME" is a long lived (daemon) executable which # Serviceguard will monitor while the package is up. # # "SERVICE_NAME", "SERVICE_FAIL_FAST_ENABLED" and "SERVICE_HALT_TIMEOUT" # specify a service for this package. #	The Servicename is the name of the Service Group within VCS. In our above examples, it is the name followed by the word group. The Service Halt Timeout is similar to a variable within VCS called OfflineTimeout. The difference is that the OfflineTimeout variable is associated with individual resources. A database may take longer to come

	# The value for "SERVICE_FAIL_FAST_ENABLED" can be either "yes" or # "no". The default is "no". If "SERVICE_FAIL_FAST_ENABLED" is set to # "yes", and the service fails, Serviceguard will halt the node on which # the service is running. # # # "SERVICE_HALT_TIMEOUT" is a number of seconds. This timeout is used # to determine the length of time the Serviceguard will wait for the # service to halt before a SIGKILL signal is sent to force the # termination of the service. In the event of a service halt, # Serviceguard will first send a SIGTERM signal to terminate the # service. If the service does not halt, Serviceguard will wait for the # specified "SERVICE_HALT_TIMEOUT", then send the SIGKILL signal to # force the service to terminate. This timeout value should be large # enough to allow all cleanup processes associated with the service to # complete. If the "SERVICE_HALT_TIMEOUT" is not specified, a zero # timeout will be assumed, meaning the cluster software will not wait at # all before sending the SIGKILL signal to halt the service. # # # Example: # SERVICE_NAME service_1a # SERVICE_FAIL_FAST_ENABLED no # SERVICE_HALT_TIMEOUT 300 # # SERVICE_NAME service_1b # SERVICE_FAIL_FAST_ENABLED no # SERVICE_HALT_TIMEOUT 300 # # SERVICE_NAME service_1c # SERVICE_FAIL_FAST_ENABLED no # SERVICE_HALT_TIMEOUT 300 # # Note: No environmental variables will be passed to the service command, this # includes the PATH variable. Absolute path names are required for the # service command definition. Default shell is /usr/bin/sh. # # Legal values for SERVICE_NAME: # Any string that starts and ends with an alphanumeric character, and # contains only alphanumeric characters, dot(.), dash(-), or underscore(_) # in between. # Maximum string length is 39 characters. # # Legal values for SERVICE_FAIL_FAST_ENABLED: yes, no. # Legal values for SERVICE_HALT_TIMEOUT: (value >= 0). SERVICE_NAME nfs1.monitor #SERVICE_FAIL_FAST_ENABLED #SERVICE_HALT_TIMEOUT	offline when compared to an IP address, which may come down instantaneously.
Storage Group	# "STORAGE_GROUP" specifies CVM specific disk group used in this package. # # WARNING: "STORAGE_GROUP" is intended to support CVM 3.5 only. This # parameter has been depreciated. It will be obsoleted in a future # Serviceguard release! For CVM 4.1 or later disk groups, please replace # it by configuring a package dependency on SG-CFS-pkg inside this package. # # Enter the names of the storage groups configured for this package. # Repeat this line as necessary for additional storage groups. # # Storage groups are only used with CVM disk groups. Neither # VxVM disk groups or LVM volume groups should be listed here. # By specifying a CVM disk group with the "STORAGE_GROUP" keyword # this package will not run until the CVM system multi node package is # running and thus the CVM shared disk groups are ready for # activation. # # Example : STORAGE_GROUP "dg01" # STORAGE_GROUP "dg02" # STORAGE_GROUP "dg03" # STORAGE_GROUP "dg04" # # Legal values for STORAGE_GROUP: # Any string that starts and ends with an alphanumeric character, and # contains only alphanumeric characters, dot(.), dash(-), or underscore(_) # in the middle. # Maximum string length is 39 characters. # #STORAGE_GROUP	VCS has resources for each LVM VG or VxVM DG. An example of this would be: LVMVolumeGroup lvg_sample1 (VolumeGroup = sample1) DiskGroup dg_sample1 (DiskGroup = sample1)
Cluster access control	# Access Control Policy Parameters. # # "USER_NAME", "USER_HOST" and "USER_ROLE" specify who can administer # this package.	cluster redhatcluster (UserNames = { admin = HopHojOlpKppNxpJom } Administrators = { admin })

	# #USER_NAME #USER_HOST #USER_ROLE	Access control and users are defined within the cluster definition with user passwords encrypted. Alternatively Secure Clusters allow using external authentication methods (see “Veritas Cluster Server User’s Guide”).
	Serviceguard configuration pkg-nfs.cntl	Veritas Cluster Server configuration main.cf

Volume Group Activation	# VOLUME GROUP ACTIVATION
	# Specify the method of activation for volume groups.
	# Leave the default ("VGCHANGE="vgchange -a y") if you want volume
	# groups activated in default mode.
	#
	# VGCHANGE="vgchange -a y"
	VGCHANGE="vgchange -a y" # Default
		VCS activates each VG individually

Volume Groups	# VOLUME GROUPS
	# Specify which volume groups are used by this package. Uncomment
	VG[0]=""
	# and fill in the name of your first volume group. You must begin with
	# VG[0], and increment the list in sequence.
	#
	# For example, if this package uses your volume groups vg01 and vg02, enter:
	# VG[0]=vg01
	# VG[1]=vg02
	#
	# Volume groups must not be set if the underlying file system is GFS.
	#
	# The volume group activation method is defined above. The filesystems
	# associated with these volume groups are specified below.
	#
	VG[0]="nfsvg"
		Here is an example of this configured for nfsvg:
		
		LVMVolumeGroup nfs_vg (
		VolumeGroup = nfsvg
		)

Volumes and Filesystems	# FILESYSTEMS
	# The only supported file systems are 'ext2', 'ext3', 'reiserfs' and 'gfs'.
	#
	# NOTE: Mixing of 'gfs' with non-gfs filesystems in the same package
	# control script is not permitted. A single package control
	# script can define either a 'gfs' filesystem or a non-gfs
	# filesystem but not both.
	#
	# The following section applies if the underlying file system is 'ext2',
	# 'ext3' or 'reiserfs'.
	#
	# The filesystems are defined as entries specifying the logical
	# volume, the mount point, the file system type, the mount,
	# umount and fsck options.
	# Each filesystem will be fsck'd prior to being mounted.
	# The filesystems will be mounted in the order specified during package
	# startup and will be unmounted in reverse order during package
	# shutdown. Ensure that volume groups referenced by the logical volume
	# definitions below are included in volume group definitions.
	#
	# Specify the filesystems which are used by this package. Uncomment
	# LV[0]=""; FS[0]=""; FS_TYPE[0]=""; FS_MOUNT_OPT[0]="";
	# FS_UMOUNT_OPT[0]=""; FS_FSCK_OPT[0]="" and fill in
	# the name of your first logical volume, filesystem, type, mount,
	# umount and fsck options for the file system.
	# You must begin with LV[0], FS[0],
	# FS_TYPE[0], FS_MOUNT_OPT[0], FS_UMOUNT_OPT[0],
	FS_FSCK_OPT[0]
	# and increment the list in sequence.
	#
	# Valid types for FS_TYPE are 'ext2', 'ext3' and 'reiserfs'.
	#
	# For example, if this package uses the following:
	# logical volume: /dev/vg01/lvol1 /dev/vg01/lvol2
	# mount point: /pkg1a /pkg1b
	# filesystem type: ext2 reiserfs
	# mount options: read/write read/write
	#
	# Then the following would be entered:
	# LV[0]=/dev/vg01/lvol1; FS[0]=/pkg1a; FS_TYPE[0]="ext2";
	# FS_MOUNT_OPT[0]="-o rw"; FS_UMOUNT_OPT[0]="";
	FS_FSCK_OPT[0]="";
	#
	# LV[1]=/dev/vg01/lvol2; FS[1]=/pkg1b; FS_TYPE[1]="reiserfs";
	# FS_MOUNT_OPT[1]="-o rw"; FS_UMOUNT_OPT[1]="";
	FS_FSCK_OPT[1]="";
	#
	#Nested mount points may also be configured
	#
	# This section applies if the underlying file system is 'gfs'
	#
	# The filesystems are defined as entries specifying the physical pool
	# device file, the mount point, the file system type and mount options.
		File System:
		
		Mount nfs_mnt (
		MountPoint = "/nfs"
		BlockDevice = "/dev/sdc1"
		FSType = ext3
		MountOpt = rw
		FsckOpt = "-y"
		)
		
		Volume Group:
		
		LVMVolumeGroup nfs_vg (
		VolumeGroup = nfsvg
		StartVolumes = 1
		)
		
		DiskReservation nfs_disk (
		Disks = { "/dev/sdc" }
		)

	# A check is performed to see if the partition is already been mounted # or not. If the partition is not mounted then it will be mounted. # # Specify the filesystems which are used by this package. Uncomment # LV[0]=""; FS[0]=""; FS_TYPE[0]=""; FS_MOUNT_OPT[0]="" and fill in # the name of your first pool, filesystem, type and mount, # options for the file system. # You must begin with LV[0], FS[0], FS_TYPE[0], # FS_MOUNT_OPT[0] and increment the list in sequence. # # Valid types for FS_TYPE are 'gfs'. # # For example, if this package uses the following: # GFS6.0 uses pool for logical volume management whereas GFS6.1 uses LVM2. # Their device name formats differ and an example for each is shown below. # Please use the appropriate one. # Pool : /dev/pool/pool1 (GFS 6.0) OR # LVM2 : /dev/mapper/vgX-lvY (GFS6.1) # mount point : /pkg1a # filesystem type : gfs # mount options : read/write # # Then the following would be entered: # LV[0]=/dev/pool/pool1; (GFS6.0) OR # LV[0]=/dev/mapper/vgX-lvY; (GFS6.1) # FS[0]=/pkg1a; FS_TYPE[0]="gfs"; # FS_MOUNT_OPT[0]="-o rw"; # LV[0]="/dev/nfsvg/nfslv"; FS[0]="/nfs"; FS_TYPE[0]="ext3"; FS_MOUNT_OPT[0]="-o rw" FS_UMOUNT_OPT[0]=""; FS_FSCK_OPT[0]=""	
File System unmount count	# FILESYSTEM UNMOUNT COUNT # Specify the number of unmount attempts for each filesystem during package # shutdown. The default is set to 1. # # This particular variable is ignored if the underlying file system # is Red Hat GFS. # FS_UMOUNT_COUNT=1	If the offline process of a resource fails then that agents clean script is run. In this case the clean script for the Mount resource will attempt to unmount forcefully.
Mount Retry Count	# FILESYSTEM MOUNT RETRY COUNT. # Specify the number of mount retrys for each filesystem. # The default is 0. During startup, if a mount point is busy # and FS_MOUNT_RETRY_COUNT is 0, package startup will fail and # the script will exit with 1. If a mount point is busy and # FS_MOUNT_RETRY_COUNT is greater than 0, the script will attempt # to kill the process(s) responsible for the busy mount point # and then mount the file system. It will attempt to kill user and # retry mount, for the number of times specified in FS_MOUNT_RETRY_COUNT. # If the mount still fails after this number of attempts, the script # will exit with 1. # NOTE: If the FS_MOUNT_RETRY_COUNT > 0, the script will execute # "fuser -kuv" to freeup busy mount point. # # FS_MOUNT_RETRY_COUNT must be set to zero (default), if the underlying # file system is of type Red Hat GFS. # FS_MOUNT_RETRY_COUNT=0	VCS has a variable for both Restarting a resource if it goes offline (RestartLimit) as well as the number of times to try to start a resource when it is first being brought online (OnlineRetryLimit). These variables are defined by the resource type or can be specified per resource.
IP Resources	# IP ADDRESSES # Specify the IP and Subnet address pairs which are used by this package. # You could specify IPv4 or IPv6 IP and subnet address pairs. # Uncomment IP[0]="" and SUBNET[0]="" and fill in the name of your first # IP and subnet address. You must begin with IP[0] and SUBNET[0] and # increment the list in sequence. # # For example, if this package uses an IP of 192.10.25.12 and a subnet of # 192.10.25.0 enter: # IP[0]=192.10.25.12 # SUBNET[0]=192.10.25.0 # (netmask=255.255.255.0) # # Hint: The subnet can be obtained by AND masking the IP address and the # netmask values from "ifconfig" command. # # For example, if this package uses an IPv6 address of 2001::1/64 # The address prefix identifies the subnet as 2001:: which is an available # subnet. # enter: # IP[0]=2001::1/64 # SUBNET[0]=2001:: # (netmask=ffff:ffff:ffff:ffff::) # # Hint: Run the "ifconfig" command and identify avaialable IPv6 subnets from # the "Global" and "Site Local" IPv6 addresses configured. # # IP/Subnet address pairs for each IP address you want to add to a subnet	IP nfs_ip (Device = eth0 Address = "192.168.1.5" NetMask = "255.255.240.0")

	# interface card. Must be set in pairs, even for IP addresses on the same # subnet. # IP[0]="192.168.1.5" SUBNET[0]="192.168.1.0"	
When will the IP be started	# HA APPLICATION SERVER # Enable or disable a High Availability application server that is used for # this package. Some examples of the HA Servers are Network File System # (NFS), Apache Web Server, and SAMBA (CIFS) Server. # # If you plan to use one of the HA server toolkits to run an application server, # you need to set the HA_APP_SERVER value to either "pre-IP" or "post-IP" in # order to enable this control script to check and run the Toolkit Interface # Script (toolkit.sh) in the package directory. The interface script will call # the toolkit main script to verify, start, and stop the server daemons. # # If you set the HA_APP_SERVER to "pre-IP", the application will be started # BEFORE adding the package IP address(es) to the system. Application servers # such as NFS and SAMBA are better to be started before the system provides # external connections (activate package IP addresses). Therefore, at the time # the clients connect to the system, the application server is # ready for service. # # If you set the HA_APP_SERVER to "post-IP", the application will be started # AFTER adding the package IP address(es) to the system. Application servers # such as Apache Web Server will check the existing IP when the server starts. # These applications will not be started if the IP has not been added to the # system. # #Uncomment one the following lines as needed: # HA_APP_SERVER="pre-IP" #HA_APP_SERVER="post-IP"	VCS defined when resources are started based on the dependency statements at the end of each service group definition in the configuration file. For example: nfs_ip requires nfs_share nfs_ip requires nfs_nic This statement tells us that the IP resource will come online after both the share resource and the NIC resource comes online.

Customer Defined section	# START OF CUSTOMER DEFINED FUNCTIONS
	
	# This function is a place holder for customer define functions.
	# You should define all actions you want to happen here, before the service is
	# started. You can create as many functions as you need.
	#
	function customer_defined_run_cmds
	{
	# ADD customer defined run commands.
	: # do nothing instruction, because a function must contain some command.
	
	test_return 51
	}
	
	# This function is a place holder for customer define functions.
	# You should define all actions you want to happen here, after the service is
	# halted.
	#
	function customer_defined_halt_cmds
	{
	# ADD customer defined halt commands.
	: # do nothing instruction, because a function must contain some command.
	
	test_return 52
	}
	
	# END OF CUSTOMER DEFINED FUNCTIONS
		Each function defined in this section should map to a defined resource to allow for each resource to be controlled (brought online/brought offline and monitored) individually. VCS has the ability to run commands in something called trigger scripts when actions trigger them, the scripts are enabled per service group.

Reference Documentation

For additional information on Veritas Cluster Server for Linux see our document repository located at:
http://sfdoccentral.symantec.com/index.html

VCS Command Line quick reference

Start VCS

```
hastart (-force) (-stale)
```

Stop VCS

```
# hastop -local [-force | -evacuate]
```
-local stops HAD on the system where you type the command.

```
# hastop -sys system_name [-force | -evacuate]
```
-sys stops had on the system you specify.

```
# hastop -all [-force]
```
-all stops had on all systems in the cluster.

Change VCS Configuration Online

```
haconf –makerw
 …make changes…
haconf –dump –makrero
```

Get Current Cluster Status

```
# hastatus -summary
```

Agent Operations

Stop and start agents manually.

```
# haagent -start agent_name -sys system_name
# haagent -stop agent_name -sys system_name
```

Add and Delete Users

Add a user with read/write access to the VCS configuration.	# hauser -add *user_name* Enter a password when prompted.
Add a user with read-only access.	# hauser -add VCSGuest Press Return when prompted for a password.
Modify a user.	# hauser -modify *user_name* Enter a new password when prompted.
Delete a user.	# hauser -delete *user_name*
Display a user. If *user_name* is not specified, all users are displayed.	# hauser -display *[user_name]*

System Operations

List systems in the cluster.	# hasys -list
Get detailed information about each system.	# hasys -display *[system_name]*
Add a system. Increase the system count in the GAB startup script.	# hasys -add *system_name*
Delete a system.	# hasys -delete *system_name*

Resource Types

List resource types.	# hatype -list
Get detailed information about a resource type.	# hatype -display *[type_name]*
List all resources of a particular type.	# hatype -resources *type_name*
Add a resource type.	# hatype -add *resource_type*
Set the value of static attributes.	# hatype -modify ...
Delete a resource type.	# hatype -delete *resource_type*

Resource Operations

List all resources	# hares -list
List a resource's dependencies.	# hares -dep *[resource_name]*
Get detailed information about a resource.	# hares -display *[resource_name]*
Add a resource.	# hares -add *resource_name resource_type service_group*
Modify the attributes of the new resource.	# hares -modify *resource_name attribute_name value*
Delete a resource, type.	# hares -delete *resource_name*
Online a resource, type.	# hares -online *resource_name* -sys *system_name*
Offline a resource, type.	# hares -offline *resource_name* -sys *system_name*
Cause a resource's agent to immediately monitor the resource on a particular system.	# hares -probe *resource_name* -sys *system_name*
Clear a faulted resource.	# hares -clear *resource_name* [-sys *system_name*]
Make a resource's attribute value local.	# hares –local *resource_name attribute_name value*
Make a resource's attribute value global.	# hares –global *resource_name attribute_name value*
Specify a dependency between two resources.	# hares -link *parent_res child_res*
Remove the dependency relationship between two resources:	# hares -unlink *parent_res child_res*

Service Group Operations

List all service groups.	# hagrp -list
List a service group's resources.	# hagrp -resources *[service_group]*
List a service group's dependencies.	# hagrp -dep *[service_group]*
Get detailed information about a service group.	# hagrp -display *[service_group]*
Start a service group and bring its resources online.	# hagrp -online *service_group* –sys *system_name*
Stop a service group and take its resources offline.	# hagrp -offline *service_group* -sys *system_name*
Switch a service group from one system to another. *(failover groups only)*	# hagrp -switch *service_group* -to *to_system*
Freeze a service group (disable onlining and offlining).	# hagrp -freeze *service_group* [-persistent]
Thaw a service group (reenable onlining and offlining).	# hagrp -unfreeze *service_group* [-persistent]
Enable a service group.	# hagrp -enable *service_group* [-sys *system_name]*
Disable a service group.	# hagrp -disable *service_group* [-sys *system_name]*
Enable all the resources in a service group.	# hagrp -enableresources *service_group*
Disable all the resources in a service group.	# hagrp -disableresources *service_group*
Specify the dependency relationship between two service groups.	# hagrp -link *parent_group child_group relationship*
Remove the dependency between two service groups.	# hagrp -unlink *parent_group child_group*

VCS Procedures

VCS Directory Structure

```
Binaries    /opt/VRTSvcs/bin
Configuration     /etc/VRTSvcs/conf/config
Logs  /var/VRTSvcs/log
```

Determine the Status of the Cluster

```
hastatus –sum
hastatus
or check out the /var/VRTSvcs/log/engine.log_A
```

To Failover the ServiceGroup from One system to another

```
hagrp –switch <SG> –to <SYSTEM>
```

To Freeze/Unfreeze the ServiceGroup

```
hagrp –freeze <SG>
hagrp –unfreeze <SG>
```

The scripts that start VCS on boot

```
/etc/rc2.d/S70llt
/etc/rc2.d/S92gab
/etc/rc3.d/S99vcs
```

To clear a faulted resource

```
First determine the reason for the fault from the log files and messages files
Second run the command:
hares –clear <RESOURCE>
```

Hastart/Hastop options

```
Hastart has to be started from each box if the cluster goes down.
If you reboot the cluster (vcs) will be started upon boot.
Hastop has two primary options (-local or –all).
When stopping the cluster you have to consider if you want just the local system within
the cluster or if the entire cluster need to have VCS stopped.
The “hastop –all –force” command will stop VCS on all nodes in the cluster but will not
stop the resources.  This allows for VCS to be shutdown without affecting the applications
that VCS is configured to manage.
```

Modifying the Cluster Config

There are three ways to modify the cluster:
1) Take all systems offline and edit the main.cf configuration file. Run "hacf –verify ."
2) Edit the cluster from the GUI while the system is up.
3) Run commands to modify the cluster while it is up.

Adding a new filesystem to the cluster

1) Create the volume from Volume Manager
2) Freeze the ServiceGroup you will be working on/modifying
3) Click to open the Cluster Configuration file
4) On the GUI click on add a resource
We will add a Mount Resource for each mounted filesystem
5) For the Mount Resource you will need the Block Device, Mount Point, and the FS Type
The Last step is to add dependencies
6) To add dependencies select the mount resource and then click on the volume resource
7) Next add all other dependencies (Mnt -> DG, if the Mount needs another mount, etc.)
8) Finally dump the cluster config to propagate the config to all other boxes
9) Then close the Cluster Config
10) When the cluster boots up and all mount points are added and are up unfreeze the ServiceGroup

reboot/init 6/shutdown commands DO failover applications

The application will come offline and the system will be rebooted. The rebooting system is executing the K10vcs rc script which contains:

$HASTOP –sysoffline

This translates to: hastop -local -evacuate -noautodisable

The "evacuate" option initiates the ServiceGroup failover. When the system comes online the ServiceGroup should be located on a different system in the cluster.

Add a user to the GUI

The cluster needs to be open to writing first, so run the command:
haconf –makerw
Next add a user with the command:
hauser –add <user>
The system will prompt you for a password.
If none is entered then the user has read-only permissions
If the added user needs more than guest permissions run the command:
haclus -modify Administrators/Operators -add <username>
hagrp -modify <grpname> Administrators/Operators -add <username>
When finished close the cluster config by running the command:
haconf –dump –makero
This command will dump the config out to all systems connected to the cluster currently,
And then close the config.

Agent Scripts

The agents rely on scripts to bring the resources online/offline/monitor.
The scripts are located in /opt/VRTSvcs/bin or /opt/VRTSagents/ha/bin directory.
Each Agent has its own directory and the online/offline/monitor/clean files
The custom agent written is located in the directory of that agent type

About Symantec

Symantec is a global leader in providing security, storage and systems management solutions to help consumers and organizations secure and manage their information-driven world. Our software and services protect against more risks at more points, more completely and efficiently, enabling confidence wherever information is used or stored.

For specific country offices and contact numbers, please visit our website.

Symantec World Headquarters
350 Ellis St.
Mountain View, CA 94043 USA
+1 (650) 527 8000
1 (800) 721 3934
www.symantec.com

02/2010 20997810