Veritas™ High Availability Agent for Documentum Content Server Installation and Configuration Guide

AIX, HP-UX, Linux, Solaris

5.1

June 2010

Veritas High Availability Agent for Content Server Installation and Configuration Guide

Agent version: 5.1.0.0

Document version: 5.1.0.0.1

Legal Notice

Symantec Corporation
350 Ellis Street
Mountain View, CA 94043

http://www.symantec.com

10 9 8 7 6 5 4 3 2 1

Technical Support

Symantec Technical Support maintains support centers globally. Technical Support's primary role is to respond to specific queries about product features and functionality. The Technical Support group also creates content for our online Knowledge Base. The Technical Support group works collaboratively with the other functional areas within Symantec to answer your questions in a timely fashion. For example, the Technical Support group works with Product Engineering and Symantec Security Response to provide alerting services and virus definition updates.

Symantec's maintenance offerings include the following:

- A range of support options that give you the flexibility to select the right amount of service for any size organization
- Telephone and/or Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers software upgrades
- Global support that is available 24 hours a day, 7 days a week
- Advance features, including Account Management Services

For information about Symantec's Maintenance Programs, you can visit our Web site at the following URL:

www.symantec.com/techsupp/

Contacting Technical Support

Customers with a current maintenance agreement may access Technical Support information at the following URL:

www.symantec.com/business/support/assistance_care.jsp

Before contacting Technical Support, make sure you have satisfied the system requirements that are listed in your product documentation. Also, you should be at the computer on which the problem occurred, in case it is necessary to replicate the problem.

When you contact Technical Support, please have the following information available:

- Product release level
- Hardware information
- Available memory, disk space, and NIC information
- Operating system

- Version and patch level
- Network topology
- Router, gateway, and IP address information
- Problem description:
  - Error messages and log files
  - Troubleshooting that was performed before contacting Symantec
  - Recent software configuration changes and network changes

Licensing and registration

If your Symantec product requires registration or a license key, access our technical support Web page at the following URL:

www.symantec.com/techsupp/

Customer service

Customer service information is available at the following URL:

www.symantec.com/techsupp/

Customer Service is available to assist with the following types of issues:

- Questions regarding product licensing or serialization
- Product registration updates, such as address or name changes
- General product information (features, language availability, local dealers)
- Latest information about product updates and upgrades
- Information about upgrade assurance and maintenance contracts
- Information about the Symantec Buying Programs
- Advice about Symantec's technical support options
- Nontechnical presales questions
- Issues that are related to CD-ROMs or manuals

Documentation feedback

Your feedback on product documentation is important to us. Send suggestions for improvements and reports on errors or omissions to clustering_docs@symantec.com. Include the title and document version (located on the second page), and chapter and section titles of the text on which you are reporting.

Maintenance agreement resources

If you want to contact Symantec regarding an existing maintenance agreement, please contact the maintenance agreement administration team for your region as follows:

Asia-Pacific and Japan	customercare_apac@symantec.com
Europe, Middle-East, and Africa	semea@symantec.com
North America and Latin America	supportsolutions@symantec.com

Contents

Chapter 1

Introducing the Veritas High Availability Agent for Content Server

This chapter includes the following topics:

About the Veritas agent for Content Server

The Veritas High Availability agents monitor specific resources within an enterprise application. They determine the status of resources and start or stop them according to external events.

The Veritas agent for Content Server provides high availability for a Content Server in a clustered environment. The Veritas High Availability agent brings specific instances of the Content Server online, monitors the instance, and brings it offline. The Veritas High Availability agent monitors the processes of the Content Server instance and shuts down the Content Server in case of a failure.

See the following Technical Support TechNote for the latest updates or software issues for this agent:

http://seer.entsupport.symantec.com/docs/282004.htm

What's new in this agent

The enhancements in this release of Content Server agent are as follows:

- Added support for HP-UX and Linux.

Supported software

The Veritas agent for Content Server supports the following software versions:

Veritas Cluster Server	■ AIX—VCS 4.0, 5.0, 5.1 ■ HP-UX—VCS 4.1, 5.0 ■ Linux—VCS 4.0, 4.1, 5.0, 5.1 ■ Solaris—VCS 4.1, 5.0, 5.1 and all intermediate Maintenance Packs of these major releases.
ACC Library	5.2.1.0 and later
Operating Systems	■ AIX 5.3, 6.1 on pSeries ■ HP-UX 11iv2 and HP-UX 11iv3 on Itanium and PA-RISC ■ Red Hat Enterprise Linux 4.0, 5.0 on Intel ■ SUSE Linux Enterprise Server 10, 11 ■ Solaris 9, 10 on SPARC
Content Server	6.5 and all intermediate minor versions of this release.

How the agent makes Content Server highly available

The Veritas agent for Content Server continuously monitors the Content Server processes to verify that they function properly.

The agent provides the following levels of application monitoring:

- Primary or Basic monitoring
  This mode has Process check and Health check monitoring options. With the default Process check option, the agent verifies that the Content Server processes are present in the process table. Process check cannot detect whether processes are in hung or stopped states.
- Secondary or Detail monitoring
  In this mode, the agent runs a utility to verify the status of Content Server. The agent detects application failure if the monitoring routine reports an

improper function of the Content Server processes. When this application failure occurs, the Content Server service group fails over to another node in the cluster.
Thus, the agent ensures high availability for Content Server.

Content Server agent functions

The agent consists of resource type declarations and agent executables. The agent executables implement online, offline, monitor, and clean operations.

Online

The online operation performs the following tasks:

- Performs the preliminary check to ensure that the Content Server instance is not online on the specified node in the cluster.
- Uses the Content Server start script `dm_start_`*`DocbaseName`* to start the Content Server instance using the name of the repository, the content server manages. The online function sources a shell script or a program that the `EnvFile` attribute specifies. The script or program ensures that the required shell environment variables are properly set before it executes the start script.
- Ensures that the Content Server instance is up and running successfully. The operation uses the wait period that the OnlineTimeout attribute specifies, to enable the Content Server instance to initialize completely before it allows the monitor function to probe the resource.

Offline

The offline operation performs the following tasks:

- Verifies that the Content Server instance is not already offline.
- Uses the Content Server stop script `dm_shutdown_`*`DocbaseName`* to stop the Content Server instance using the name of the repository the Content Server manages. The offline function also sources a shell script or a program that the `EnvFile` attribute specifies. The script or program ensures that the required shell environment variables are properly set before it executes the stop script.
- Ensures that the Content Server instance is given enough time to go offline successfully. The operation uses a wait period that the OfflineTimeout attribute specifies, to allow the Content Server instance to complete the offline sequence before it allows further probing of the resource.

Monitor

The monitor function monitors the state of the Content Server instance running on all nodes within the cluster.

The monitor operation performs following tasks:

- The first level check scans the system process table and searches the processes that must be running for Content Server instance. If the first level check does not find these processes running on the node, the check exits immediately, and reports the Content Server instance as offline.
- If the SecondLevelMonitor attribute is set to greater than 0, the monitor function performs a second-level check to determine the status of the Content Server instance. The Content Server installation provides the java program. The second-level check runs the java program provided by Content Server installation to ensure that the processes are truly available for Content Server instance.
- Depending upon the MonitorProgram attribute, the monitor function can perform a customized check using a user-supplied monitoring utility.
  See "Executing a customized monitoring program" on page 29.

Clean

In case of a failure or after an unsuccessful attempt to be online or offline, the clean function removes any Content Server processes remaining in the system.

The clean operation performs following tasks:

- Attempts to gracefully shut down the Content Server instance.
- If a graceful shutdown fails, the clean function looks for all the processes running for the Content Server instance, and cleans the processes by killing them.

Chapter 2

Installing and configuring Content Server for high availability

This chapter includes the following topics:

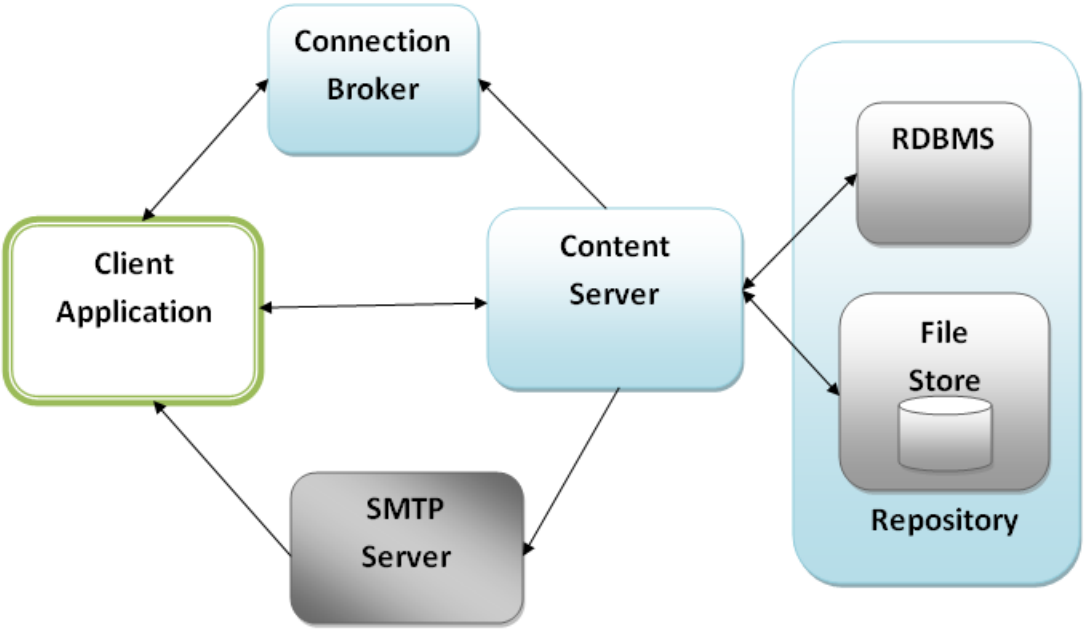
About Content Server

Content Servers are the processes that provide client access to the repository. They receive queries from clients in the form of DFC methods or DQL statements and make a call to the underlying RDBMS or the file directories. Every repository must have at least one active server. If a repository does not have an active server, then users cannot access that repository.

When a server is started, the resulting server process is called a parent server. Each time a client asks for a repository connection through a parent server, the parent server spawns another server process to service that client. These spawned server processes are called session servers. They are active until the session is active. When the session terminates, the session servers also terminate. The number of session servers that can be spawned from a parent server are configurable.

The Content Server depends on the Connection Broker for its functioning. The connection broker is intermediary between the client and the server when a client wants a repository connection. If a server is not known to even one connection broker, the clients cannot connect to the repository associated with the server. Each server regularly projects connection information to at least one connection broker. When a client wants to connect to a content server repository, it contacts the connection broker and requests for the connection information for the Content Server repository. The connection broker sends back the IP address for the host where the Content Server resides and the port number that the Content Server uses. The client application uses the information to open a connection to Content Server.

Figure 2-1 A simple deployment topology



In simple configuration, connection broker can be installed and started on the Content Server host or the Content Server can project one or more connection brokers that are located on a different host.

Uniquely identifying Content Server instances

For multiple Content Server instances running concurrently on a single node, the Veritas agent must be able to uniquely identify each of the Content Server instance on that system. Each Content Server instance has a unique configuration or an initialization file. The Veritas agent uses the `InitFile` attribute value to identify the Content Server instance uniquely.

Differentiating the Content Server instances is important to identify each Content Server uniquely. When the Veritas agent kills the processes of a non-responsive or failed Content Server instance, in absence of an unique `InitFile` for each Content Server instance, the Veritas agent may kill processes for more than one Content Server instance during a clean operation.

About installing Content Server for high availability

Install the Content Server on each node in a VCS cluster.

Ensure that you perform the installation such that the data directory of the content server installation or the data part of the repository is on the shared storage and all the content server installation can access the data when required.

When installing Content Server, ensure that the user name, UNIX uid, group name, and UNIX gid for the Documentum user is the same on all the nodes.

The user and the group must be local and not Network Information Service (NIS and NIS+) users.

For more details refer to the product documentation.

About configuring Content Server for high availability

To configure the Content Server instance for high availability ensure that you install it on all the nodes in the cluster and associate it to a virtual host/IP.

In addition, store the repository (managed by content server) data on a shared storage.

During the service group configuration ensure that you create a service group dependency between the Content Server and Connection Broker agent service groups. The dependency type between the Connection Broker service group and service group containing the resource for Content Server must be 'Online Global Soft'.

In a typical configuration when the Connection Broker goes offline the Content Server cannot establish any new client connection. However, the existing client

connections remain unaffected. Thus, the service group containing the Content Server should not fault or fail over in case the Connection Broker service group faults. The Connection Broker can be running on any node and the Content Server configuration would be aware of the Connection Broker hostname and the port to which it has to broadcast the information. Once the Connection Broker is online again, the Content Server broadcasts the information to the Connection Broker and the session is resumed. This eliminates the need to restart the Content Server.

Hence the dependency type between two service group is 'Online Global Soft' with the Content Server group as the parent and the Connection Broker group as the child.

Configuring the Content Server for high availability

This section provides the information about the tasks you must perform to configure Content Server for high availability.

Synchronizing accounts and services

Ensure that you synchoronize accounts and services in the following ways:

- Synchronize the Documentum user accounts user name, UNIX uid, group name, and UNIX gid across all nodes in the cluster.
- The /etc/services entries should be consistent on all cluster nodes.

Removing physical host dependencies

Perform the following tasks to remove the physical host dependencies:

- Add *host* parameter to `server.ini` file under [SERVER_STARTUP] section if it does not exist (`$DOCUMENTUM/dba/confing/$DocbaseName/server.ini`).
  `host =` *content server virtual hostname*
- Modify [DOCBROKER_PROJECTION_TARGET] and [DOCBROKER_PROJECTION_TARGET_n] sections in the server.ini file to have correct connection broker information.
  `host =` *connection broker virtual hostname*
  `port =` *connection broker port number*

Chapter 3

Installing, upgrading, and removing the agent for Content Server

This chapter includes the following topics:

Before you install the Veritas agent for Content Server

For VCS, do the following:

- Install and configure Veritas Cluster Server.
  For more information on installing and configuring Veritas Cluster Server, refer to the *Veritas Cluster Server Installation Guide*.
- Install the latest version of ACC Library.
  To install or update the ACC Library package, locate the library and related documentation on the agentpack disc.
  See "Installing the ACC library" on page 18.

About the ACC library

The operations of a VCS agent depend on a set of Perl modules known as the ACC library. The library must be installed on each system in the cluster that runs the agent. The ACC library contains common, reusable functions that perform tasks, such as process identification, logging, and system calls.

The ACC library installation package is included within each agent's software distribution media (tar file or CD). Instructions to install or remove the ACC library on a single system in the cluster are given in the following sections. The instructions assume that the agent's tar file has already been extracted or that you are working from the agent's installation CD.

Installing the ACC library

Install the ACC library on each system in the cluster that runs an agent that depends on the ACC library.

To install the ACC library

1. Log in as superuser.
2. Download the complete agent pack tarball from FileConnect site:

   https://fileconnect.symantec.com/

   or the individual ACCLib tarball from the Symantec Veritas Operations Services (VOS) site:

   https://vos.symantec.com/home
3. If you downloaded the complete Agent Pack tarball, navigate to the directory containing the package for the platform running in your environment.

AIX	*cd1*/aix/vcs/application/acc_library/*version*_library/pkgs
HP-UX	*cd1*/hpux/generic/vcs/application/acc_library/*version*_library/pkgs
Linux	*cd1*/linux/generic/vcs/application/acc_library/*version*_library/rpms
Solaris	*cd1*/solaris/*dist_arch*/vcs/application/acc_library/*version*_library/pkgs where *dist_arch* is sol_sparc

4 If you downloaded the individual ACCLib tarball, navigate to the pkgs directory (for AIX, HP-UX, and Solaris), or rpms directory (for Linux).

5 Install the package. Enter **Yes** if asked to confirm overwriting of files in the existing package.

AIX
```
# installp -ac -d VRTSacclib.bff VRTSacclib
```

HP-UX
```
# swinstall -s 'pwd' VRTSacclib
```

Linux
```
# rpm -i \
VRTSacclib-VersionNumber-GA_GENERIC.noarch.rpm
```

Solaris
```
# pkgadd -d VRTSacclib.pkg
```

Installing the agent in a VCS environment

Install the agent for Content Server on each node in the cluster.

Note: The agent package VRTSvcsdctm includes the Veritas agents for Content Server and Connection Broker. So, the following procedure to install the agent for ContentServer also installs the agent for ConnectionBroker.

To install the agent in a VCS environment

1 Download the complete agent pack tarball from FileConnect site:

https://fileconnect.symantec.com/

Alternatively,

Download the individual agent tarball from the Symantec Veritas Operations Services (VOS) site:

https://vos.symantec.com/home

2 Uncompress the file to a temporary location, say /tmp.

3 If you downloaded the complete Agent Pack tarball, navigate to the directory containing the package for the platform running in your environment.

AIX	`cd1/aix/vcs/application/documentum_agent/` `*vcs_version*/*version*_agent/pkgs`
HP-UX	`cd1/hpux/generic/vcs/application/documentum_agent/` `*vcs_version*/*version*_agent/pkgs`
Linux	`cd1/linux/generic/vcs/application/documentum_agent/` `*vcs_version*/*version*_agent/rpms`

Solaris	`cd1/solaris/*dist_arch*/vcs/application/documentum_agent/`
	`*vcs_version*/*version*_agent/pkgs`
	where, *dist_arch* is sol_sparc

If you downloaded the individual agent tarball, navigate to the pkgs directory (for AIX, HP-UX, and Solaris), or rpms directory (for Linux).

4 Log in as superuser.

5 Install the package.

AIX	`# installp -ac -d VRTSvcsdctm.rte.bff VRTSvcsdctm.rte`
HP-UX	`# swinstall -s 'pwd' VRTSvcsdctm`
Linux	`# rpm -ihv \` `VRTSvcsdctm-*AgentVersion*-GA_GENERIC.noarch.rpm`
Solaris	`# pkgadd -d . VRTSvcsdctm`

Removing the agent in a VCS environment

You must uninstall the agent for Content Server from a cluster while the cluster is active.

Warning: The agent package VRTSvcsdctm includes the Veritas agents for Content Server and Connection Broker. So, the following procedure to remove the agent for ContentServer also removes the agent for ConnectionBroker.

To uninstall the agent in a VCS environment

1. Log in as a superuser.
2. Set the cluster configuration mode to read/write by typing the following command from any node in the cluster:

   ```
   # haconf -makerw
   ```

3. Remove all Content Server resources from the cluster. Use the following command to verify that all resources have been removed:

   ```
   # hares -list Type=ContentServer
   ```

4. Remove the agent type from the cluster configuration by typing the following command from any node in the cluster:

   ```
   # hatype -delete ContentServer
   ```

   Removing the agent's type file from the cluster removes the include statement for the agent from the main.cf file, but the agent's type file is not removed from the cluster configuration directory. You can remove the agent's type file later from the cluster configuration directory.
5. Save these changes. Then set the cluster configuration mode to read-only by typing the following command from any node in the cluster:

   ```
   # haconf -dump -makero
   ```

6. Use the platform's native software management program to remove the agent for Content Server from each node in the cluster.

   Execute the following command to uninstall the agent:

AIX	`# installp -u VRTSvcsdctm.rte`
HP-UX	`# swremove VRTSvcsdctm`
Linux	`# rpm -e VRTSvcsdctm`
Solaris	`# pkgrm VRTSvcsdctm`

Removing the ACC library

Perform the following steps to remove the ACC library.

To remove the ACC library

1. Ensure that all agents that use ACC library are removed.
2. Run the following command to remove the ACC library package.

AIX	`# installp -u VRTSacclib`
HP-UX	`# swremove VRTSacclib`
Linux	`# rpm -e VRTSacclib`
Solaris	`# pkgrm VRTSacclib`

Chapter 4

Configuring the agent for Content Server

This chapter includes the following topics:

About configuring the Veritas agent for Content Server

After installing the Veritas agent for Content Server, you must import the agent type configuration file. After importing this file, you can create and configure a Content Server resource. Before you configure a resource, review the attributes table that describes the resource type and its attributes.

To view the sample agent type definition and service groups configuration.

See "About sample configurations for the agent for Content Server" on page 41.

Importing the agent types files in a VCS environment

To use the agent for Content Server, you must import the agent types file into the cluster.

To import the agent types file using the Veritas Cluster Server graphical user interface

1. Start the Veritas Cluster Manager and connect to the cluster on which the agent is installed.
2. Click **File > Import Types**.
3. In the Import Types dialog box, select the following file:

VCS 4.x	■ AIX ■ HP-UX ■ Linux ■ Solaris	`/etc/VRTSvcs/conf/sample_ContentServer/DocumentumTypes.cf`
VCS 5.x	■ AIX ■ HP-UX ■ Linux	`/etc/VRTSagents/ha/conf/ContentServer/DocumentumTypes.cf`
VCS 5.0	Solaris	`/etc/VRTSagents/ha/conf/ContentServer/DocumentumTypes50.cf`
VCS 5.1	Solaris	`/etc/VRTSagents/ha/conf/ContentServer/DocumentumTypes51.cf`

4. Click **Import**.
5. Save the VCS configuration.

   The ContentServer agent type is now imported to the VCS engine.

Note: The Documentum.cf file contains the agent type definition for ContentServer and ConncetionBroker. Hence, the above procedure will import the agent type definition for both ContentServer and ConnectionBroker agent.

You can now create Content Server resources. For additional information about using the VCS GUI, refer to the *Veritas Cluster Server User's Guide*.

To import the agent types file using the Veritas Cluster Server command line interface (CLI), perform the following steps.

1. Log on to any one of the systems in the cluster as the superuser.
2. Create a temporary directory.

```
# mkdir ./temp

# cd ./temp
```

3 Copy the sample file Types.cf from the following location:

VCS 4.x	■ AIX ■ HP-UX ■ Linux ■ Solaris	`/etc/VRTSvcs/conf/sample_ContentServer/ContentServerTypes.cf`
VCS 5.x	■ AIX ■ HP-UX ■ Linux	`/etc/VRTSagents/ha/conf/ContentServer/ContentServerTypes.cf`
VCS 5.0	■ Solaris	`/etc/VRTSagents/ha/conf/ContentServer/DocumentumTypes50.cf`
VCS 5.1	■ Solaris	`/etc/VRTSagents/ha/conf/ContentServer/DocumentumTypes51.cf`

4 Create a dummy main.cf file:

```
# echo 'include "ContentServerTypes.cf"' > main.cf
```

5 Create the Content Server resource type as follows:

```
# hacf -verify .

# haconf -makerw

# sh main.cmd

# haconf -dump
```

The ContentServer agent type is now imported to the VCS engine.

You can now create Content Server resources. For additional information about using the VCS CLI, refer to the *Veritas Cluster Server User's Guide*.

Documentum Content Server agent attributes

Refer to the required attributes and optional attributes while configuring the agent for ContentServer.

Table 4-1 lists the required attributes for the ContentServer agent.

Table 4-1 Required attributes

Required attributes	Description
DocbaseName	Specifies the name of the docbase repository which the content server manages. Type and Dimension: string-scaler Default: "" Example: REPCVR
DMBase	Specifies the absolute path of the directory where the content server scripts `dm_start_`*`DocbaseName`* reside. Type and Dimension: string-scaler Default: "" Example: /documentum/dba
InitFile	Specifies the configuration or Initialization file for the Content Server instance. The Veritas agent uses this attribute value to uniquely identify the running Content Server instance. Type and Dimension: string-scaler Default: "" Example: /documentum/dba/config/REPCVR/server.ini
DMUser	Specifies the user name that the Veritas Agent uses to execute the programs for managing a content server. The user name must be synchronized across the systems in the cluster. The user name must resolve to the same UID and have the same default shell on each system in the cluster. The Veritas Agent entry points use the getpwnam (3c) function call to obtain UNIX user attributes. Hence, the user can be defined locally or can be defined in a common repository (NIS, NIS+, or LDAP). If the user is defined to a repository, the agent will fail if the access to the repository fails. The supported shell environments are: ksh, sh, and csh. Type and Dimension: string-scaler Default: "" Example: cvradm

Table 4-1 Required attributes *(continued)*

Required attributes	Description
EnvFile	Specifies the absolute path to the file that must be sourced with the UNIX shell. Source this file to set the environment before executing Content Server scripts for online, offline, monitor, and clean operations. The shell environments supported are: ksh, sh, and csh. **Note:** Ensure that the syntax of this file is in accordance with the user shell that the DMUser attribute specifies. Review the information about how to generate environments file for Documentum ContentServer. See "Generating the environments file for Content Server" on page 34. Type and Dimension: string-scaler Default: "/dev/null" Example: /documentum /envfile
ResLogLevel	Specifies the logging detail performed by the agent for the resource. The valid values are as follows: ■ ERROR: Only logs error messages. ■ WARN: Logs above plus warning messages. ■ INFO: Logs above plus warning messages. ■ TRACE: Logs above plus trace messages. TRACE is very verbose and should only be used during initial configuration or for troubleshooting and diagnostic operations. Type and Dimension: string-scaler Default: INFO Example: TRACE

Table 4-2 lists the optional attributes for the ContentServer agent.

Table 4-2 Optional attributes

Optional attribute	Description
SecondLevelMonitor	Used to enable second-level monitoring. Second-level monitoring is a deeper, more thorough state check of the Content Server. The numeric value specifies how often the monitoring routines must run. 0 means never run the second-level monitoring routines, 1 means run routines every monitor interval, 2 means run routines every second monitor interval. This interpretation may be extended to other values. **Note:** Exercise caution while setting SecondLevelMonitor to large numbers. For example, if the MonitorInterval is set to 60 seconds and the SecondLevelMonitor is set to 100, then the second level check is executed every 100 minutes, which may not be as often as intended. For maximum flexibility, no upper limit is defined for SecondLevelMonitor. Also, verify the second level monitoring utility before enabling second level monitor. See "Verifying Second Level Monitor command" on page 39. Type and Dimension: integer-scaler Default: 0 Example: 5
MonitorProgram	Specifies the absolute path of an external, the user supplied monitor executable. See "Executing a customized monitoring program" on page 29. Type and Dimension: string-scaler Default: "" Example: /documenytum/myMonitor.sh
JavaHome	Specifies the absolute path of the Documentum java bin directory. The value of this attribute is used to run the SecondLevelMonitor java program. Type and Dimension: string-scaler Default: "" Example: /documentum/shared/java/1.5.0_12

Executing a customized monitoring program

The monitor function executes a custom monitor program to perform an additional Content Server state check. The monitor function executes the utility specified in the MonitorProgram attribute, if:

- The specified utility is a valid executable file.
- The first level process check indicates that the Content Server instance is online.
- The SecondLevelMonitor attribute is either set to 0 or 1, and the second level check indicates that the Content Server instance is online.
- The SecondLevelMonitor attribute is set to greater than 1, but the second level check is deferred for this monitoring cycle.

The monitor function interprets the utility exit code as follows:

110 or 0	ContentServer instance is online
110 or 1	ContentServer instance is offline
99	ContentServer instance is unknown
Any other value	ContentServer instance is unknown

To ensure that the customized utility is always available to the agent, Symantec recommends storing the file in a shared directory that is available on an online node.

Chapter 5

Configuring the service groups for Content Server

This chapter includes the following topics:

About configuring service groups for Content Server

Configuring the Content Server service group involves creating the ContentServer service group, its resources, and defining attribute values for the configured resources. You must have administrator privileges to create and configure a service group.

You can configure the service groups using one of the following:

- The Cluster Manager (Java console)
- The command-line

See "Configuring service groups for Content Server" on page 32.

Before configuring the service groups for Content Server

Before you configure the Content Server service group, you must:

- Verify that VCS is installed and configured on all nodes in the cluster where you will configure the service group.
  Refer to the *Veritas Cluster Server Installation Guide* for more information.
- Verify that Content Server is installed and configured identically on all nodes in the cluster.
  See "About installing Content Server for high availability" on page 15.
  See "About configuring Content Server for high availability " on page 15.
- Verify that the Veritas agent for Content Server is installed on all nodes in the cluster.
  See "Installing the agent in a VCS environment" on page 19.
- Verify that the type definition for the Veritas agent for Content Server is imported into the VCS engine.
  See "Importing the agent types files in a VCS environment" on page 23.

Configuring service groups for Content Server

While setting up a cluster, you must ensure that the cluster has some spare capacity to handle the Content Server failover scenarios.

The cluster should be able to provide application failover by encapsulating the resources required for an application into a service group. A service group is a virtualized application that can switch between the cluster nodes. It contains a set of dependent resources, such as disk groups, disk volumes, file systems, IP addresses, NIC cards, and dependent application processes. It also includes logic about the dependencies between the application components.

These service groups should thus be configured such that the cluster can start, stop, monitor, and switch the service groups between the nodes, depending upon the server faults or resource faults. An administrator should also be proactively able to move a service group between cluster nodes to perform preventative maintenance or apply patches.

Perform the following steps to add a service group for Content Server

1 Create a service group for Content Server.

```
# hagrp -add DCM652-CS
```

For more details on creating a service group refer to, *Veritas Cluster Server User's Guide*

2 Modify the SystemList attribute for the group, to add systems.

For example,

```
# hagrp -modify DCM652-CS SystemList systemA 0 systemB 1
```

3 Create resources for NIC and IP in the service group.

For example,

```
# hares -add DCM652-CS_nic NIC DCM652-CS

# hares -DCM652-CS_ip IP DCM652-CS
```

For more details on creating and modifying resource attributes for NIC, IP, DiskGroup, Volume, and Mount refer to, *Bundled Agents Reference Guide*

4 Create links between the resources.

For example,

```
# hares -link DCM652-CS_ip DCM652-CS_nic
```

5 If you have not installed the data directory of the repository on shared file system then follow the steps 6 onwards, otherwise, directly go to step 9 and continue further.

6 Create a separate file system for Content Server on shared disk.

7 Copy the contents of the data directory of the Content Server to this shared file system.

8 Delete the contents of the data directory and then create a link between the data directory and the directory on the shared disk, on which you copied the contents of that data directory.

9 Add the file system to respective agent service group using the Mount, DiskGroup, and Volume resources.

Create Mount and DiskGroup resources.

For example,

```
# hares -add DCM652-CS_dg DiskGroup DCM652-CS

# hares -DCM652-CS_mnt Mount DCM652-CS
```

Based on the Content Server instance you cluster, modify the resource attributes of the Mount and DiskGroup resources.

10 Create links between the Mount and DiskGroup resources.

For example,

```
# hares -link DCM652-CS_mnt DCM652-CS_dg
```

11 Create the resource for the Content Server.

For example,

```
# hares -add DCM652-CS_cs ContentServer DCM652-CS
```

Based on the Content Server instance you cluster, modify the resource attributes.

See "Documentum Content Server agent attributes" on page 25.

12 Create resource dependencies for ContentServer resource.

The ContentServer resource depends on the IP and Mount resources.

```
# hares -link DCM652-CS_cs DCM652-CS_ip
```

```
# hares -link DCM652-CS_cs DCM652-CS_mnt
```

13 Verify the final resource dependencies for DCM652-CS server group.

For example,

```
# hares -dep
```

```
Group       Parent          Child
DCM652-CS   DCM652-CS_cs    DCM652-CS _ip
DCM652-CS   DCM652-CS_cs    DCM652-CS _mnt
DCM652-CS   DCM652-CS_ip    DCM652-CS_nic
DCM652-CS   DCM652-CS_mnt   DCM652-CS_dg
```

Generating the environments file for Content Server

To generate the environments file for Content Server

1 Login as Documentum user using the following command.

```
su - dmadmin
```

2 Capture the environment with the following command.

```
env > /home/dmadmin/dmadmin.env
```

3 Modify the file according to the Documentum user shell environment.

For example, if the generated file contains environments for bash shell and Documentum user shell is C shell, convert the file to C shell environments.

- Edit the dmadmin.env file to add string 'setenv' at the beginning of each line.

- Replace the '=' with space " " in the file.

4 Copy the dmadmin.env file to shared directory and use it as the Content Server instance environments file in EnvFile attribute. Ensure that the permissions are set properly for user Documentum user.

```
chmod 755 dmadmin.env
```

Note: Before generating the EnvFile, verify the successful execution of start, stop, and second level monitor command with Documentum user environment.

Chapter 6

Troubleshooting the agent for Content Server

This chapter includes the following topics:

Using the correct software and operating system versions

Ensure that no issues arise due to incorrect software and operating system versions. For the correct versions of operating system and software to be installed on the resource systems:

See "Supported software" on page 10.

Meeting prerequisites

Before installing the agent for Content Server, double check that you meet the prerequisites.

For example, you must install the ACC library on VCS before installing the agent for Content Server.

See "Before you install the Veritas agent for Content Server" on page 17.

Configuring Content Server resources

Before using a Content Server resource, ensure that you configure the resource properly. For a list of attributes used to configure all Content Server resources, refer to the agent attributes.

See "Documentum Content Server agent attributes" on page 25.

Starting the Content Server instance outside a cluster

If you face problems while working with a resource, you must disable the resource within the cluster framework. A disabled resource is not under the control of the cluster framework, and so you can test the Content Server instance independent of the cluster framework. Refer to the cluster documentation for information about disabling a resource.

You can then restart the Content Server instance outside the cluster framework.

Note: Use the same parameters that the resource attributes define within the cluster framework while restarting the resource outside the cluster framework.

A sample procedure to start a Content Server instance outside the cluster framework, is illustrated as follows.

To restart the Content Server outside the VCS framework

1. Log in to the Content Server node as an DMUser.

   ```
   # su - DMUser
   ```

2. Source the environment file.

   ```
   #. EnvFile
   ```

3. Start the Content Server.

   ```
   # DMBase/dm_start_DocbaseName
   ```

If the Content Server instance works properly outside the cluster framework, attempt to implement the Content Server instance within the cluster framework.

Verifying Second Level Monitor command

If you have enabled Second Level Monitoring and are facing problems with the Content Server agent resource, verify whether the second level monitor command is working properly outside the cluster control. To ensure proper working, you must disable the resource within the cluster framework.

A disabled resource is not under the control of the cluster framework, and so you can test the Content Server instance independent of the cluster framework. Refer to the cluster documentation for information about disabling a resource. Then you can verify the second level monitor command for the Content Server instance outside the cluster framework.

A sample procedure is illustrated as follows.

To verify the second level monitor command for a Content Server instance outside the cluster framework

1 Log in to theContent Server node as a DMUser.

```
# su – DMUser
```

2 Source the environment file.

```
# . EnvFile
```

3 Execute the second level command

```
# JavaHome/bin/java
com.documentum.server.impl.utils.ContentServerStatus -docbase_name
DocbaseName -user_name DMUser
```

If the command works properly outside the cluster framework, attempt to implement the Content Server within the cluster framework.

Reviewing error log files

If you face problems while using Content Server or the agent for Content Server, use the log files described in this section to investigate the problems.

Using Content Server log files

If Content Server instance is facing problems, access the Content Server log files to diagnose the problem. The Content Server log files are located in the *DMBase*/log/ directory.

Reviewing cluster log files

In case of problems while using the agent for Content Server, you can access the engine log file for more information about a particular resource. The engine log file is located at /var/VRTSvcs/log/engine_A.log.

You can also access the ContentServer agent log file for more detailed information. The agent log file is located at /var/VRTSvcs/log/ContentServer_A.log

Using trace level logging

The ResLogLevel attribute controls the level of logging that is written in a cluster log file for each Content Server resource. You can set this attribute to TRACE, which enables very detailed and verbose logging.

If you set ResLogLevel to TRACE, a very high volume of messages are produced. Symantec recommends that you localize the ResLogLevel attribute for a particular resource.

To localize ResLogLevel attribute for a resource

1. Identify the resource for which you want to enable detailed logging.
2. Localize the ResLogLevel attribute for the identified resource:

   ```
   # hares -local Resource_Name ResLogLevel
   ```

3. Set the ResLogLevel attribute to TRACE for the identified resource:

   ```
   # hares -modify Resource_Name ResLogLevel TRACE -sys SysA
   ```

4. Note the time before you begin to operate the identified resource.
5. Test the identified resource. The function reproduces the problem that you are attempting to diagnose.
6. Note the time when the problem is reproduced.
7. Set the ResLogLevel attribute back to INFO for the identified resource:

   ```
   # hares -modify Resource_Name ResLogLevel INFO -sys SysA
   ```

8. Review the contents of the log file. Use the time noted in Step 4 and Step 6 to diagnose the problem.

   You can also contact Symantec support for more help.

Appendix A

Sample Configurations

This appendix includes the following topics:

About sample configurations for the agent for Content Server

The sample configuration graphically depicts the resource types, resources, and resource dependencies within the service group. Review these dependencies carefully before configuring the agent for Content Server. For more information about these resource types, see the *Veritas Cluster Server Bundled Agents Reference Guide*.

Sample agent type definition

This section lists the sample agent type definition files for Content Server agent on different versions of VCS.

For VCS 4.x

```
type ContentServer (
static str ArgList[] = { ResLogLevel, State, IState, DocbaseName, DMBase,
InitFile, DMUser, EnvFile, JavaHome, MonitorProgram, SecondLevelMonitor }
         str ResLogLevel = INFO
```

```
        str DocbaseName
        str DMBase
        str InitFile
        str DMUser
        str EnvFile = "/dev/null"
        str JavaHome
        str MonitorProgram
        int SecondLevelMonitor = 0
)
```

For VCS 5.x

```
type ContentServer (
        static boolean AEPTimeout = 1
        static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
        static str AgentDirectory = "/opt/VRTSagents/ha/bin/ContentServer"
static str ArgList[] = { ResLogLevel, State, IState, DocbaseName, DMBase,
InitFile, DMUser, EnvFile, JavaHome, MonitorProgram, SecondLevelMonitor }
        str ResLogLevel = INFO
        str DocbaseName
        str DMBase
        str InitFile
        str DMUser
        str EnvFile = "/dev/null"
        str JavaHome
        str MonitorProgram
        int SecondLevelMonitor = 0
)
```

Sample configuration

This section provides a sample configuration for Content Server agent. The sample configuration depicts a graphical view of the resource types, resources, and resource dependencies within the service group.

```
include "types.cf"
include "DocumentumTypes.cf"
cluster cluster1 (
        UserNames = { admin = dlmElgLimHmmKumGlj }
        ClusterAddress = "110.120.162.128"
        Administrators = { admin }
        UseFence = SCSI3
        HacliUserLevel = COMMANDROOT
```

```
        )
system systemA  (
       )
system systemB  (
       )
system systemC  (
       )

group DCM652-CS (
        SystemList = { systemA = 0,  systemB =1 }
        )
ContentServer DCM652-CS_cs (
                Critical = 0
                DocbaseName = REPCVR
                DMBase = "/documentum/dba"
                InitFile = "/documentum/dba/config/REPCVR/server.ini"
                DMUser = cvradm
                EnvFile = "/documentum/env.sh"
                JavaHome = "/documentum/shared/java/1.5.0_12"
                MonitorProgram =/documentum/monitor.sh"
                )
DiskGroup DCM652-CS_dg (
                DiskGroup = dcm652cvr_dg
                )
IP DCM652-CS_ip (
                Device = bge0
                Address = "110.120.62.18"
                NetMask = "255.255.255.0"
                )
Mount DCM652-CVRCS1_mnt (
                MountPoint = "/documentum/data"
                BlockDevice = "/dev/vx/dsk/dcm652cvr_dg/dcm652cvr_vol"
                FSType = vxfs
                FsckOpt = "-y"
                )
NIC DCM652-CS_nic (
        Device = bge0
        Network Type = ether
   )
DCM652CS_cs requires DCM652-CS_ip
DCM652-CS_cs requires DCM652-CS_mnt
DCM652-CS_ip requires DCM652-CS_nic
DCM652-CS_mnt requires DCM652-CS1_dg
```

```
// resource dependency tree
//
//      group DCM652-CS
//      {
//      ContentServer DCM652-CS_cs
//          {
//          IP DCM652-CS_ip
//              {
//              NIC DCM652-CS_nic
//              }
//          Mount DCM652-CS_mnt
//              {
//              DiskGroup DCM652-CS_dg
//              }
//          }
//      }
```

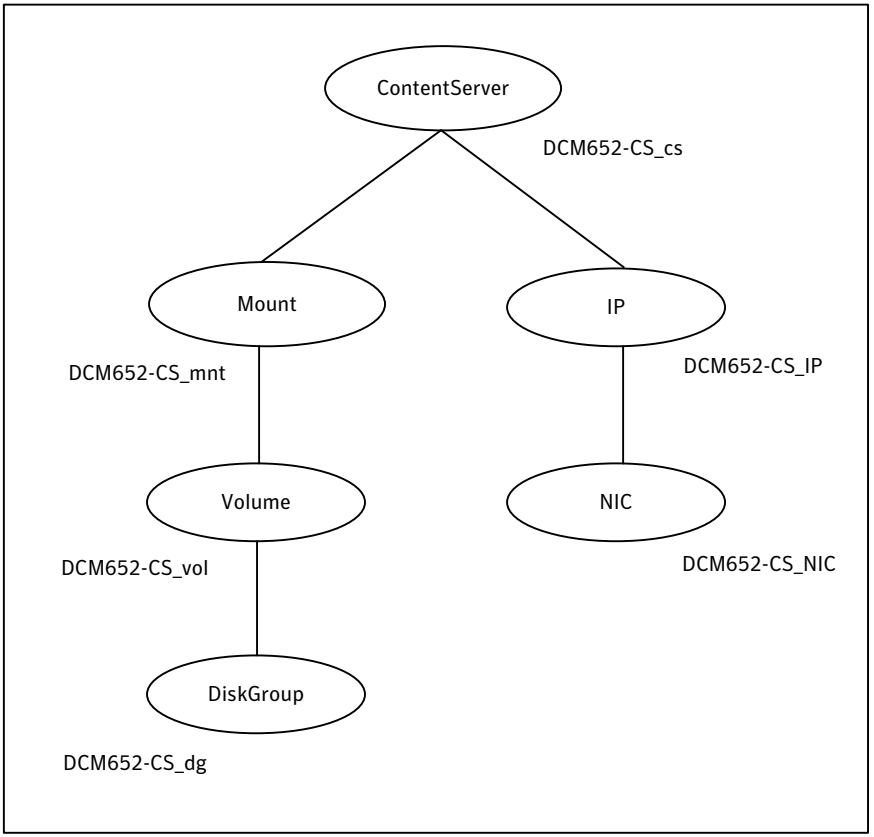
Sample service group configuration

The service group configuration in a cluster depends on some common characteristics that must be part of the configuration design.

The Content Server instance should have a separate virtual IP address assigned to facilitate network transparency.

Figure A-1 shows a sample service group configuration for ContentServer instance.

Figure A-1 Service group configuration for ContentServer instance

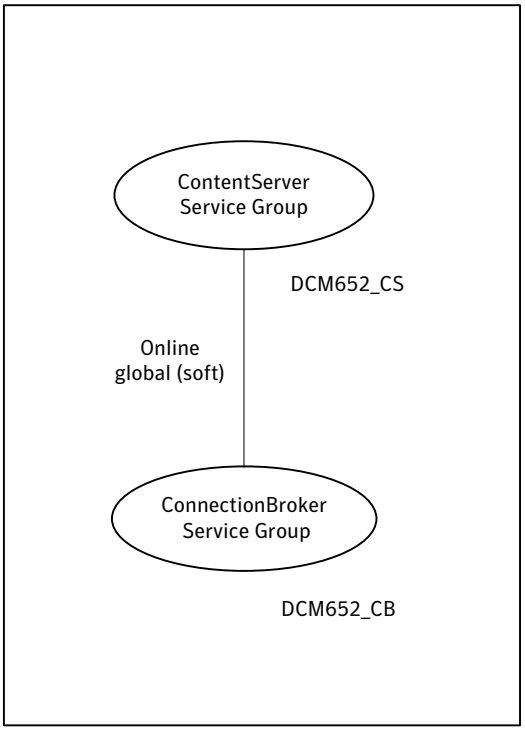


Sample service group dependency for Content Server and Connection Broker

This section includes service groups that show the group dependency between ContentServer service group and ConnectionBroker service group.

Figure A-2 shows the sample service group dependency for Documentum.

Figure A-2 Sample service group dependency



Index

U