Veritas™ High Availability Agent for Documentum Connection Broker Installation and Configuration Guide

AIX, HP-UX, Linux, Solaris

5.1

June 2010

Veritas High Availability Agent for Connection Broker 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 Connection Broker

This chapter includes the following topics:

About the Veritas agent for Connection Broker

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 Connection Broker provides high availability for a Connection Broker in a clustered environment. The Veritas High Availability agent brings specific instances of the Connection Broker online, monitors the instance, and brings it offline. The Veritas High Availability agent monitors the processes of the Connection Broker instance and shuts down the Connection Broker 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 Connection Broker agent are as follows:

- Added support for HP-UX and Linux.

Supported software

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
Connection Broker	6.5 and all intermediate minor versions of this release.

How the agent makes Connection Broker highly available

The Veritas agent for Connection Broker continuously monitors the Connection Broker 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 Connection Broker 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 Connection Broker. The agent detects application failure if the monitoring routine reports an improper function of the Connection Broker processes. When this application failure occurs, the Connection Broker service group fails over to another node in the cluster.
  Thus, the agent ensures high availability for Connection Broker.

Connection Broker 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 Connection Broker instance is not online on the specified node in the cluster.
- Uses the Connection Broker start script `dm_launch_CBName` to start the Connection Broker instance using the name of Connection Broker and Connection Broker initialization file. 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 Connection Broker instance is up and running successfully. The operation uses the wait period that the OnlineTimeout attribute specifies, to enable the Connection Broker 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 Connection Broker instance is not already offline.
- Uses the Connection Broker stop script `dm_stop_CBName` to stop the Connection Broker instance using the name of the repository the Connection Broker manages. The offline function also sources a shell script or a program that the `EnvFile` attribute specifies. This script or program ensures that the required shell environment variables are properly set before it executes the stop script.
- Ensures that the Connection Broker instance is given enough time to go offline successfully. The operation uses a wait period that the OfflineTimeout attribute

specifies, to allow the Connection Broker instance to complete the offline sequence before it allows further probing of the resource.

Monitor

The monitor function monitors the state of the Connection Broker 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 Connection Broker instance. If the first level check does not find these processes running on the node, the check exits immediately, and reports the Connection Broker 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 Connection Broker instance. The second-level check runs the `dmqdocbroker` utility or program, provide by Connection Broker installation to ensure that the processes are truly available for Connection Broker 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 31.

Clean

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

The clean operation performs following tasks:

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

Chapter 2

Installing and configuring Connection Broker for high availability

This chapter includes the following topics:

About Connection Broker

The Documentum connection broker is a process that provides client sessions with connection information. To establish a connection, a client session must know where to find a server that accesses the requested repository. When a client session is opened, the client contacts the connection broker and requests the information it needs to connect with a server for the requested repository. The connection brokers that can handle a client's connection request are defined in the client's dfc.properties file.

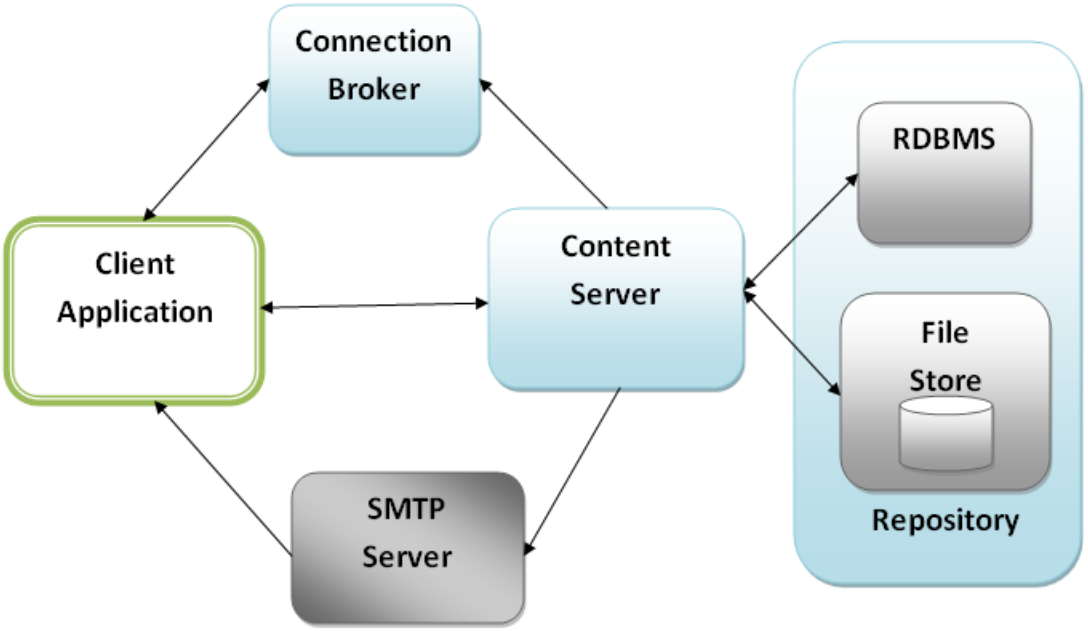
The connection broker sends back the IP address for the host where the Content Server resides and the port number that the Content Server uses. If there are multiple servers for the repository, the connection broker returns connection information for all of them. The client session uses the information to choose a server and open the connection. Clients can request a connection through a

particular server, any server on a particular host, or a particular server on a specified host. Each connection broker has information about servers and the repositories they access.

A connection broker keeps track of when it last heard from a server and when it expects to hear from that server. If it does not receive a broadcast from a server at the expected time, it sets the server's status to presumed down. It keeps the entry for a non-broadcasting server for a certain number of hours. At the end of the allotted time (the keep entry interval), if the connection broker has not heard from the server, it removes the server from its registry. When a system administrator shuts down a server gracefully, the server sends out a message informing the connection brokers that it is going down. The connection brokers that hear that message set the server's status to stopped. Later, when the server is restarted, it rebroadcasts its information and the connection brokers update their entries for the server and reset its status.

A client is anything requesting a connection to a server. For example, a client can be Webtop, an external application, or a user working with Documentum Administrator. In each case, at the start of a new client session, a connection must be established with a repository. To establish the initial connection, the client application first sends a message to a connection broker asking for the service information that it needs to make the requested connection. Each client session has a default connection broker. Generally, clients also have backup connection brokers to ensure continuous service. The default connection brokers and backup connection brokers for clients are defined in the dfc.properties file. Each EMC Documentum web-based client application contains a dfc.properties file that is packaged with the application's WAR file.

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 Connection Broker instances

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

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

About installing Connection Broker for high availability

Install the Connection Broker on each node in a VCS cluster.

When installing Connection Broker, 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 Connection Broker for high availability

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

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 Connection Broker for high availability

This section provides the information about the tasks you must perform to configure Connection Broker 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:

- Modify the host parameter in the script `dm_launch_`*`CBName`* and pass the parameter to the start command `dmdocbroker`.
  `host =` *connection broker virtual hostname*
  Example:

  ```
  host=CBHost

   ./dmdocbroker -host $host -port 1489

  -init_file/documentum/dba/$CBName.ini $@ >> $tlogfile 2>&1 &
  ```

- Edit the configuration file and modify the parameter in the script.
  Edit the connection broker configuration file (`$DOCUMENTUM/dba/`*`$CBName`*`.ini`) to set the virtual host and the port on which the connection broker listens. Change the host, port, and service from the [DOCBROKER_CONFIGURATION] section, present in the configuration file.
  `host =` *Connection Broker virtual hostname*
  `service =` *service_name*
  `port =` *port_number*
- Modify the –T parameter option to the `dmshutdown` command in dm_stop_*CBName* script.
  Example:

  ```
  ./dmshutdown docbroker -B –T virtual_hostname -P -N1489 $@
  ```

Chapter 3

Installing, upgrading, and removing the agent for Connection Broker

This chapter includes the following topics:

Before you install the Veritas agent for Connection Broker

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 20.

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 Connection Broker on each node in the cluster.

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

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 Connection Broker from a cluster while the cluster is active.

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

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 ConnectionBroker resources from the cluster. Use the following command to verify that all resources have been removed:

   ```
   # hares -list Type=ConnectionBroker
   ```

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

   ```
   # hatype -delete ConnectionBroker
   ```

   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 Connection Broker 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 Connection Broker

This chapter includes the following topics:

About configuring the Veritas agent for Connection Broker

After installing the Veritas agent for Connection Broker, you must import the agent type configuration file. After importing this file, you can create and configure a ConnectionBroker 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 Connection Broker" on page 41.

Importing the agent types files in a VCS environment

To use the agent for Connection Broker, 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_ConnectionBroker/DocumentumTypes.cf`
VCS 5.x	■ AIX ■ HP-UX ■ Linux	`/etc/VRTSagents/ha/conf/ConnectionBroker/DocumentumTypes.cf`
VCS 5.0	Solaris	`/etc/VRTSagents/ha/conf/ConnectionBroker/DocumentumTypes50.cf`
VCS 5.1	Solaris	`/etc/VRTSagents/ha/conf/ConnectionBroker/DocumentumTypes51.cf`

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

   The ConnectionBroker 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 ConnectionBroker 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_ConnectionBroker/ConnectionBrokerTypes.cf`
VCS 5.x	■ AIX ■ HP-UX ■ Linux	`/etc/VRTSagents/ha/conf/ConnectionBroker/ConnectionBrokerTypes.cf`
VCS 5.0	■ Solaris	`/etc/VRTSagents/ha/conf/ConnectionBroker/DocumentumTypes50.cf`
VCS 5.1	■ Solaris	`/etc/VRTSagents/ha/conf/ConnectionBroker/DocumentumTypes51.cf`

4 Create a dummy main.cf file:

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

5 Create the Connection Broker resource type as follows:

```
# hacf -verify .

# haconf -makerw

# sh main.cmd

# haconf -dump
```

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

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

Documentum Connection Broker agent attributes

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

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

Table 4-1 Required attributes

Required attributes	Description
InitFile	Specifies the configuration or Initialization file for the Connection Broker instance. The attribute value is used as the parameter to start script and also in identifying the Connection Broker instance uniquely. Type and Dimension: string-scaler Default: "" Example: /documentum/dba/DBroker.ini
CBName	Specifies the connection broker name. The attribute value is used to uniquely identify, start, and stop the connection broker. Type and Dimension: string-scaler Default: "" Example: DBroker
DMBase	Specifies the absolute path of the directory where the Connection Broker binaries `dm_launch_CBName, dm_stop_CBName` reside. Type and Dimension: string-scaler Default: "" /Documentum/dba
Password	Specifies the encrypted password for the Documentum user. Refer to the VCS documentation for more information about VCSEncrypt. Type and Dimension: string-scaler Default: "" Example: codOeoFidleiFig

Table 4-1 Required attributes *(continued)*

Required attributes	Description
DMUser	Specifies the user name that the Veritas agent uses to execute the connection broker commands. 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
EnvFile	Specifices the absolute path to the file that must be sourced with the UNIX shell. Source this file to set the environment before executing Connection Broker 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 ConnectionBroker. See "Generating the environments file for Connection Broker" on page 36. Type and Dimension: string-scaler Default: "/dev/null" Example: /documentum/envfile

Table 4-1 Required attributes *(continued)*

Required attributes	Description
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 ConnectionBroker agent.

Table 4-2 Optional attributes

Optional attributes	Description
SecondLevelMonitor	Used to enable second-level monitoring. Second-level monitoring is a deeper, more thorough state check of the Connection Broker. 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

Table 4-2 Optional attributes *(continued)*

Optional attributes	Description
MonitorProgram	Specifies the absolute path of an external, the user supplied monitor executable. See "Executing a customized monitoring program" on page 31. Type and Dimension: string-scaler Default: "" Example: /documentum/myMonitor.sh
DMHome	Specifies the absolute path of the Documentum installation directory. The Veritas Agent uses the attribute to locate the `dmqdocbroker` command. Type and Dimension: string-scaler Default: "" Example:/documentum/product/6.5/
CBHost	Specifies the virtual host name by which the connection broker is running. Type and Dimension: string-scaler Default: "" Example: DBHost
CBPort	Specifies the port number by which the connection broker is listening. Type and Dimension: integer-scaler Default; "1489" Example: 1489

Executing a customized monitoring program

The monitor function executes a custom monitor program to perform an additional Connection Broker 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 Connection Broker instance is online.

- The SecondLevelMonitor attribute is either set to 0 or 1, and the second level check indicates that the Connection Broker 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	ConnectionBroker server instance is online
100 or 1	ConnectionBroker server instance is offline
99	ConnectionBroker server instance is unknown
Any other value	ConnectionBroker server 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 Connection Broker

This chapter includes the following topics:

About configuring service groups for Connection Broker

Configuring the Connection Broker service group involves creating the ConnectionBroker 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 Connection Broker" on page 34.

Before configuring the service groups for Connection Broker

Before you configure the ConnectionBroker 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 Connection Broker is installed and configured identically on all nodes in the cluster.
  See "About installing Connection Broker for high availability" on page 16.
  See "About configuring Connection Broker for high availability " on page 16.
- Verify that the Veritas agent for Connection Broker is installed on all nodes in the cluster.
  See "Installing the agent in a VCS environment" on page 21.
- Verify that the type definition for the Veritas agent for Connection Broker is imported into the VCS engine.
  See "Importing the agent types files in a VCS environment" on page 25.

Configuring service groups for Connection Broker

While setting up a cluster, you must ensure that the cluster has some spare capacity to handle the Connection Broker 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 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 Connection Broker

1. Create a service group for Connection Broker.

   ```
   # hagrp -add DCM652-CB
   ```

   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-CB SystemList systemA 0 systemB 1
   ```

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

   For example,

   ```
   # hares -add DCM652-CB_nic NIC DCM652-CB
   ```

   ```
   # hares -DCM652-CB_ip IP DCM652-CB
   ```

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

4. Create links between the resources.

   For example,

   ```
   # hares -link DCM652-CB_ip DCM652-CB_nic
   ```

5. Create the resource for the Connection Broker.

   For example,

   ```
   # hares -add DCM652-CB_broker ConnectionBroker DCM652-CB
   ```

   Based on the Connection Broker instance you cluster, modify the resource attributes.

   See "Documentum Connection Broker agent attributes" on page 27.

6 Create resource dependencies for Connection Broker resource.

The Connection Broker resource depends on the IP and Mount resources.

```
# hares -link DCM652-CB_broker DCM652-CB_ip
```

7 Verify the final resource dependencies for DCM652-CB server group.

For example,

```
# hares -dep

Group       Parent              Child
DCM652-CB   DCM652-CB_broker   DCM652-CB_ip
DCM652-CB   DCM652-CB_ip       DCM652-CB_nic
```

Generating the environments file for Connection Broker

To generate the environments file for ConnectionBroker

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 ConnectionBroker 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 Connection Broker

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 Connection Broker, double check that you meet the prerequisites.

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

See "Before you install the Veritas agent for Connection Broker" on page 19.

Configuring ConnectionBroker resources

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

See "Documentum Connection Broker agent attributes" on page 27.

Starting the ConnectionBroker 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 ConnectionBroker instance independent of the cluster framework. Refer to the cluster documentation for information about disabling a resource.

You can then restart the ConnectionBroker 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 Connection Broker instance outside the cluster framework, is illustrated as follows.

To restart the ConnectionBroker outside the VCS framework

1. Log in to the ConnectionBroker node as an DMUser.

   ```
   # su - DMUser
   ```

2. Source the environment file.

   ```
   #. EnvFile
   ```

3. Start the Connection Broker.

   ```
   # DMBase/dm_start_CBName
   ```

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

Verifying Second Level Monitor command

If you have enabled Second Level Monitoring and are facing problems with the Connection Broker agent resource, verify whether the second level monitor command `dmqdocbroker` 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 ConnectionBroker 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 ConnectionBroker instance outside the cluster framework.

A sample procedure is illustrated as follows.

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

1. Log in to theConnectionBroker node as a DMUser.

   ```
   # su – DMUser
   ```

2. Source the environment file.

   ```
   # . EnvFile
   ```

3. Execute the second level command

   ```
   # DMHome/bin/dmqdocbroker –t CBHost –p CBPort –c ping
   ```

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

Reviewing error log files

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

Using ConnectionBroker log files

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

Reviewing cluster log files

In case of problems while using the agent for Connection Broker, 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 ConnectionBroker agent log file for more detailed information. The agent log file is located at /var/VRTSvcs/log/ConnectionBroker_A.log

Using trace level logging

The ResLogLevel attribute controls the level of logging that is written in a cluster log file for each ConnectionBroker 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 Connection Broker

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 Connection Broker. 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 Connection Broker agent on different versions of VCS.

For VCS 4.x

```
type ConnectionBroker (
static str ArgList[] = { ResLogLevel, State, IState, CBName, DMHome,
DMBase, CBHost, CBPort, InitFile, DMUser, Password, EnvFile,
MonitorProgram, SecondLevelMonitor }
```

```
         str ResLogLevel = INFO
         str CBName
         str DMHome
         str DMBase
         str CBHost
         int CBPort = 1489
        str InitFile
        str DMUser
         str Password
        str EnvFile = "/dev/null"
         str MonitorProgram
         int SecondLevelMonitor = 0
 )
```

For VCS 5.x

```
type ConnectionBroker (
         static boolean AEPTimeout = 1
         static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
         static str AgentDirectory = "/opt/VRTSagents/ha/bin/ConnectionBroker"
static str ArgList[] = { ResLogLevel, State, IState, CBName, DMHome,
DMBase, CBHost, CBPort, InitFile, DMUser, Password, EnvFile,
MonitorProgram, SecondLevelMonitor }
         str ResLogLevel = INFO
         str CBName
         str DMHome
         str DMBase
         str CBHost
         int CBPort = 1489
         str InitFile
         str DMUser
         str Password
         str EnvFile = "/dev/null"
         str MonitorProgram
         int SecondLevelMonitor = 0
 )
```

Sample configuration

This section provides a sample configuration for Connection Broker 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-CB (
        SystemList = { systemA = 0,  systemB =1 }
        )
ConnectionBroker DCM652-CB_broker (
       Critical = 0
       CBName = Docbroker
       DMHome = "/documentum/product/6.5"
       DMBase = "/documentum/dba"
       CBHost = dcm652host1
       InitFile = "/documentum/dba/Docbroker.ini"
       DMUser = cvradm
       Password = ESKqHSh
       EnvFile = "/home/cvradm/.profile"
       MonitorProgram = "/documentum/monitor.sh
)

IP DCM652-CB_ip (
       Critical = 0
       Device = bge0
       Address = "110.229.222.194"
       NetMask = "255.255.255.0"
)
NIC DCM652-CB_nic (
        Device = bge0
        Network Type = ether
   )
DCM652-CB_broker requires DCM652-CB_ip
```

```
DCM652-CB_ip requires DCM652-CB_nic

 // resource dependency tree
//
//      group DCM652-CB
//      {
//      ConnectionBroker DCM652-CB_broker
//          {
//          IP DCM652-CB_ip
//              {
//              NIC DCM652-CB_nic
//              }
//          }
//      }
```

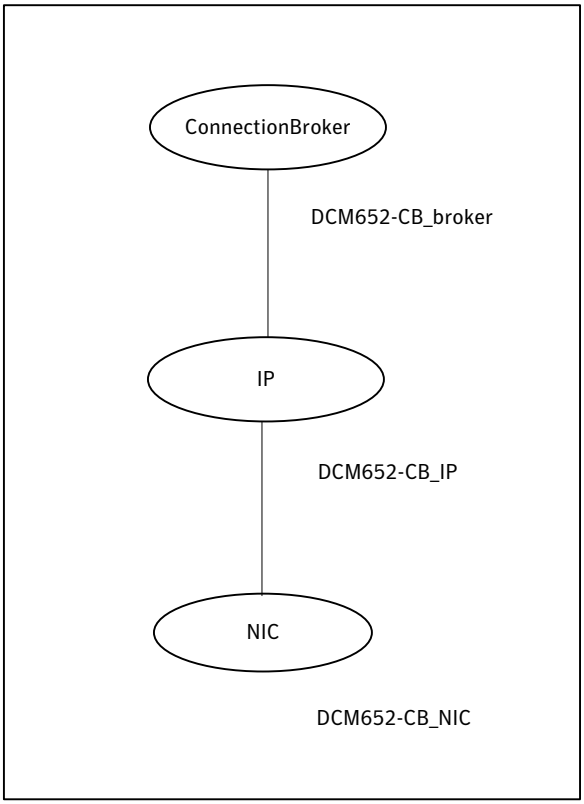
Sample service group configuration

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

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

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

Figure A-1 Service group configuration for Connection Broker instance

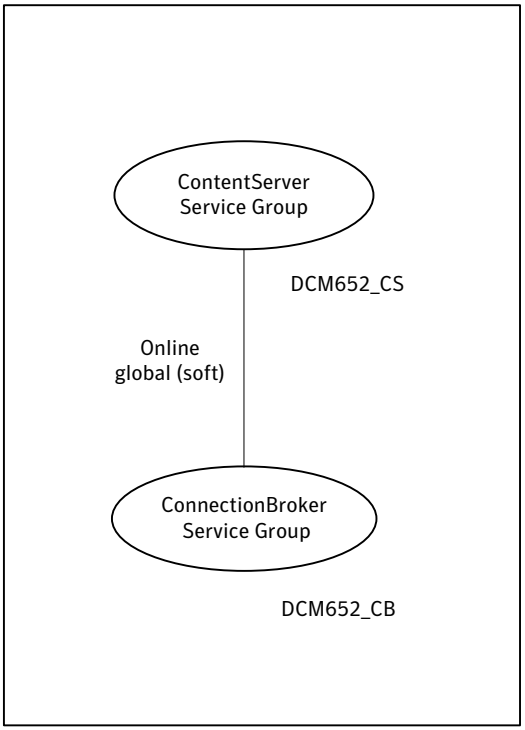


Sample service group dependency for Content Server and Connection Broker

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