Veritas™ High Availability Agent for MySQL Installation and Configuration Guide

AIX, HP-UX, Linux, Solaris

5.1

September 2010

Symantec™

Veritas High Availability Agent for MySQL Installation and Configuration Guide

Agent version: 5.1.0.1

Document version: 5.1.0.1.1

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 MySQL

This chapter includes the following topics:

About the Veritas agent for MySQL

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 High Availability Agent for MySQL provides high availability for all the MySQL servers in a cluster.

This guide assumes that the user understands the primary components and basic functionality of a cluster. The guide also assumes that the user has a basic understanding of the MySQL applications architecture and the administration tasks required to start, stop, monitor, and operate these components.

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

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

Features of the Veritas Agent for MySQL

- Enables the validation of attributes based on entry points.
  Validates the attributes in each entry point, before the actual data processing starts. Hence, the code is robust.
- First Failure Data Capture (FFDC)
  In case of a fault, the agent generates a huge volume of debug logs that enable troubleshooting of the fault.
- Fast First Level Monitor (FFLM)
  Maintains PID files based on search patterns to expedite the monitoring process.
- Supports external user-supplied monitor utilities
  In addition to the built-in monitoring logic, user-specified monitor utilities can be plugged-in. This enables the administrator to customize the monitoring of the application.
- Delay entry point
  The agent intelligently delays the first monitor after online for slow initializing applications.

Supported software

The Veritas agent for MySQL supports the following software versions in a VCS environment:

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 4.0, 4.1, 5.0, 5.1 and all intermediate Maintenance Packs of these major releases. **Note:** n the software disc, no separate agent is provided for VCS 4.0 and 5.1 on Linux and Solaris. To use the agent for VCS 4.0 and 5.1 on both these platforms, use the agent provided for VCS 4.1 and 5.0 respectively. Similarly, on AIX, to use the agent for VCS 5.1, use the agent provided for VCS 5.0
Operating Systems	■ AIX 5.1, 5.2, 5.3, 6.1 on pSeries ■ HP-UX 11i v2, 11i v3 on Itanium and PA-RISC ■ Linux Red Hat Enterprise 4.0, 5.0 on Intel and ppc64 ■ SUSE Linux Enterprise Server 9, 10, 11 on Intel and ppc64 ■ Solaris 8, 9, 10 on SPARC including the zones ■ Solaris 10 x64
ACC Library	5.2.2.0
MySQL	5.0 and 5.1 and all intermediate minor versions of these releases.

How the agent makes MySQL highly available

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

MySQL agent functions

The operations or functions that the Veritas High Availability Agent for MySQL can perform are described as follows:

Online

The online function performs the following tasks:

- Verifies that the required attributes are set correctly.
- Verifies that the MySQL Server instance is not already online. If the instance is online, the online operation exits immediately
- If any MySQL processes remain, the operation kills these processes using the user name associated with the specific resource.
- Attempts to start the MySQL server instance with the command:

  ```
  $ BaseDir/bin/mysqld_safe --defaults-file=MyCnf \
   --datadir=DataDir --user=MySQLUser
  ```

  The command always gets executed in the context of MySQLUser, specifying the MySQL configuration file, if specified by the MyCnf agent attribute.
- Checks if the server has started up completely.
- Gives the control back to HAD.

Offline

The offline function performs the following tasks:

- Verifies that the required attributes are set correctly.
- Verifies that the MySQL Server instance is not offline.
- If the instance is already offline, the operation verifies if any processes belonging to this MySQL resource, exist.
- Attempts to stop the MySQL server instance with the command:

  ```
  $ BaseDir/bin/mysqladmin --user=MySQLAdmin \
  --password=MySQLAdminPasswd shutdown
  ```

  The command always gets executed in the context of MySQLUser.
  Then the offline operation kills any existing processes that belong to this MySQL server instance. Gives the control back to HAD.

Monitor

The monitor function monitors the states of the MySQL Servers on all nodes within the cluster. The operation performs the following tasks:

- The monitor function conducts a first level check to determine that the MySQL Server processes, are running on the system in the cluster. If the first level check does not find these processes running on the node, the check exits immediately, and reports the instance as OFFLINE.

  Note: The agent sets the cluster MySQL type level attribute, ToleranceLimit to 1. This ensures that the application gets an opportunity to restart a failed mysqld instance, before the agent flags the instance OFFLINE, to initiate a failover.

- If the SecondLevelMonitor attribute is set to greater than 0, the monitor operation conducts a second level check.
- The agent uses a connect(3c) method on the IP address specified by the HostName agent attribute to check for the MySQL server to listen to the port defined by the Port attribute.
- The agent then uses the monitor command to verify that the MySQL server is really up.

  ```
  $ BaseDir/bin/mysqladmin --user=MySQLAdmin \
  --password=MySQLAdminPasswd status
  ```

  The command is executed in the context of the MySQLUser.
- Depending upon the MonitorProgram attribute, the monitor operation can perform a customized check using a user-supplied monitoring utility. Please refer to the agent attributes for more details regarding this attribute.
  the section called “MySQL agent attributes”

Clean

In case of a failure or after an unsuccessful attempt to online or offline a MySQL Server instance, the clean operation performs the following tasks:

- Attempts to gracefully shut down the MySQL server instance with the command:

  ```
  $ BaseDir/bin/mysqladmin --user=MySQLAdmin \
  --password=MySQLAdminPasswd shutdown
  ```

  The command always gets executed in the context of MySQLUser.

- The clean operation kills any remaining process pertaining to this MySQL instance.
- Gives the control back to HAD.

Setting up MySQL in a VCS cluster

Follow the steps below to set up MySQL in a cluster:

- Set up a VCS cluster.
- Install and configure MySQL for High Availability.
- Install the Veritas High Availability agent for MySQL.
  See "Installing the agent in a VCS environment" on page 23.
- Configure the service groups for MySQL.
  See "About configuring service groups for MySQL" on page 35.

Chapter 2

Installing and configuring MySQL for high availability

This chapter includes the following topics:

About MySQL

MySQL is a relational database management system (RDBMS). The MySQL software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.

Installing the MySQL instance

Review the following section while installing a MySQL database instance:

- Specifying shared disk for storing the database

Specifying shared disk for storing the database

Specifying the database on shared storage ensures that the database is available on the failover node before the application is brought online. The database directory is specified using the DataDir agent attribute. This attribute must be identical to that specified in the configuration file (my.cnf) the database instance uses for starting up.

For information on the DataDir attribute, review the Agent attributes section.

See "MySQL agent attributes" on page 29.

Setting MySQL parameters after installation

It is possible to host multiple instances of MySQL database on the same physical node by using different database configuration files. Each instance of the MySQL database can be customized, and is then registered with the agent using the MyCnf agent attribute.

For information on the MyCnf attribute, review the Agent attributes section.

See "MySQL agent attributes" on page 29.

Review the following sections while customizing the database configuration file:

- See "Configuring the MySQL base directory and database directory" on page 16.
- See "Configuring virtual IP addresses" on page 16.
- See "Configuring Ports" on page 17.
- See "Configuring the MySQL database user" on page 17.

Configuring the MySQL base directory and database directory

Ensure that each database instance manages a unique database directory, specified by the "datadir" configuration parameter. The base (or installation) directory, specified by the "basedir" configuration parameter may be shared across multiple instances of the database server. These values need to be registered with the agent using the BaseDir and DataDir agent attributes.

Review the information on the BaseDir and DataDir agent attributes.

See "MySQL agent attributes" on page 29.

Configuring virtual IP addresses

To ensure that the database is available to clients from all failover nodes, it must be hosted non-promiscuously. Use a virtual hostname which gets resolved to a

unique IP address on all failover nodes of the cluster for specifying the HostName agent attribute. Also ensure that this is specified in the database configuration file (my.cnf) via the “bind-address” configuration parameter.

An IP address should be used as its value as a workaround for a bug in some versions of MySQL as reported by:

http://bugs.mysql.com/bug.php?id=28516

Review the information on the HostName agent attribute.

See “MySQL agent attributes” on page 29.

Configuring Ports

To ensure that multiple instances can be hosted on the same failover node, the HostName/Port pair combination has to be unique. MySQL server by default listens on port 3306. This can be changed using the “port” configuration parameter.

Configuring the MySQL database user

It is recommended to use a non-root user while starting a MySQL database. This is the UNIX user owning the database directory and its files. The value for this attribute should be identical to the “user” database configuration parameter, if specified in the database configuration file (my.cnf), and should be registered with the agent using the MySQLUser agent attribute.

The following is an excerpt from a typical MySQL configuration file (my.cnf) that is used to start a database instance.

```
# The following options will be passed to all MySQL clients
[client]
# password       = your_password
port            = 3306
socket          = /tmp/mysql.sock
# Here follows entries for some specific programs
# The MySQL server
[mysqld]
user            = mysql
basedir         = /usr/local/mysql
datadir         = /db/bbmas/data
pid-file        = /db/bbmas/data/pidfile.bbmas
port            = 3306
socket          = /tmp/mysql.sock
tmpdir          = /var/tmp
.. truncated ..
```

```
# Specify the bind address
bind-address    = 10.209.72.140
```

Adding a dedicated database administrator with shutdown privileges only

It is strongly recommended that you create a dedicated database administrator with privileges only to shutdown a particular instance of the database, locally. Do not use the default “root” database administrator that has unrestricted database privileges, as the agent does not need them.

To add a dedicated database administrator *MySQLAdmin* at the mysql prompt do the following:

```
mysql> create user 'MySQLAdmin'@'localhost' identified by 'XXXXXX' ;
Query OK, 0 rows affected (0.00 sec)
mysql> create user 'MySQLAdmin'@'127.0.0.1' identified by 'XXXXXX' ;
Query OK, 0 rows affected (0.00 sec)
mysql> grant shutdown on *.* to 'MySQLAdmin'@'localhost' ;
Query OK, 0 rows affected (0.00 sec)
mysql> grant shutdown on *.* to 'MySQLAdmin'@'l27.0.0.1' ;
Query OK, 0 rows affected (0.00 sec)
mysql> quit
```

This assumes that the session owner has grant access to add a database user and assign privileges for database shutdown to that user.

Ensure that you can shutdown the database instance using this database user:

```
$ BaseDir/bin/mysqladmin --user=MySQLAdmin --password=XXXXXX shutdown
```

Where *MySQLAdmin* is the database administrator being created and registered with the agent, and *XXXXXX* is the password being set for this administrator, which is encrypted and specified using the MySQLAdminPasswd agent attribute.

For information on MySQLAdmin and MySQLAdminPasswd attributes, review the Agent attributes section.

See “MySQL agent attributes” on page 29.

Virtualizing MySQL

To ensure that your MySQL machine can function properly on any node of the cluster, you need to virtualize all the parameters that could be dependent on a particular node.

Review the following basic notes for virtualization:

Host names	When installing and configuring the MySQL machine, ensure that you enter the virtual host name associated with the IP address used to configure the IP resource. This ensures that if the application needs to be migrated, you are not tied down by the physical IP address given to the MySQL machine.
Path names	Ensure that your application gets installed on a shared disk so that it is not constrained by anything that is local to the node. If this is not possible every time, make sure that the local data is available on each configured node.

Running multiple instances of MySQL on a single node

The agent supports hosting multiple instances of the MySQL database server on a single physical node. To do this:

Add the environment variables `MYSQL_UNIX_PORT` and `MYSQL_TCP_PORT` to the environment file being used with the agent via the EnvFile agent attribute. The following is an excerpt from a typical environment file for the Bourne shell:

```
MYSQL_UNIX_PORT      = socket; export MYSQL_UNIX_PORT
MYSQL_TCP_PORT       = port; export MYSQL_TCP_PORT
```

Where,

socket	The value of the 'socket' database parameter under the [mysqld] section of the configuration file
port	The value of the 'port' database parameter under the [mysqld] section of the configuration file

For an excerpt of a typical MySQL configuration file: See "Configuring the MySQL database user" on page 17.

Hence, for the current example, the sample environment file will be:

```
MYSQL_UNIX_PORT      = /tmp/mysql.sock; export MYSQL_UNIX_PORT
MYSQL_TCP_PORT       = 3306; export MYSQL_TCP_PORT
```

Note: This procedure is also valid when multiple instances are not being hosted.

Chapter 3

Installing, upgrading, and removing the agent for MySQL

This chapter includes the following topics:

Before you install the Veritas agent for MySQL

You must install the Veritas agent for MySQL on all the systems that will host an MySQL service group.

Ensure that you meet the following prerequisites to install the agent for MySQL.

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 22.

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.

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.

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 or sol_x64.

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 MySQL on each node in the cluster.

To install the agent in a VCS environment

1 Download the complete agent pack tarball from the FileConnect site:

https://fileconnect.symantec.com/

Alternatively,

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

https://vos.symantec.com/agents

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/database/mysql_agent/*vcs_version*/*version*_agent/pkgs
HP-UX	cd1/hpux/generic/vcs/database/mysql_agent/*vcs_version*/*version*_agent/pkgs
Linux	cd1/linux/generic/vcs/database/mysql_agent/*vcs_version*/*version*_agent/rpms
Solaris	cd1/solaris/*dist_arch*/vcs/database/mysql_agent/*vcs_version*/*version*_agent/pkgs where, *dist_arch* is sol_x64 or 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 VRTSmysql.rte.bff VRTSmysql.rte`
HP-UX	`# swinstall -s 'pwd' VRTSmysql`
Linux	`# rpm -ihv \` `VRTSmysql-`*`AgentVersion`*`-GA_GENERIC.noarch.rpm`
Solaris	`# pkgadd -d . VRTSmysql`

Removing the agent in a VCS environment

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

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

```
# hares -list Type=MySQL
```

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

```
# hatype -delete MySQL
```

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 MySQL from each node in the cluster.

Execute the following command to uninstall the agent:

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

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 MySQL

This chapter includes the following topics:

About configuring the Veritas agent for MySQL

After installing the Veritas agent for MySQL, you must import the agent type configuration file. After importing this file, you can create and configure a MySQL 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 MySQL" on page 43.

Importing the agent types files in a VCS environment

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 5.x	■ AIX ■ HP-UX ■ Linux	`/etc/VRTSagents/ha/conf/MySQL/MySQLTypes.cf`

4 Click **Import**.

5 Save the VCS configuration.

You can now create MySQL 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_MySQL/MySQLTypes.cf`
VCS 5.x	■ AIX ■ HP-UX ■ Linux	`/etc/VRTSagents/ha/conf/MySQL/MySQLTypes.cf`
VCS 5.0	■ Solaris SPARC and x64	`/etc/VRTSagents/ha/conf/MySQL/MySQLTypes50.cf`

4 Create a dummy main.cf file:

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

5 Create the MySQL resource type as follows:

```
# hacf -verify .

# haconf -makerw

# sh main.cmd

# haconf -dump
```

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

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

MySQL agent attributes

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

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

Table 4-1 Required attributes

Attribute	Description
ResLogLevel	Specifies the logging detail that the agent performs for the resource. The valid values are as follows: ■ ERROR: Only logs error messages. ■ WARN: Logs above plus warning messages. ■ INFO: Logs above plus informational messages ■ TRACE: Logs above plus trace messages. TRACE is very verbose and should be used only during initial configuration or for troubleshooting and diagnostic operations. Default Value: INFO Example: INFO

Table 4-1 Required attributes *(continued)*

Attribute	Description
MySQLAdmin	The administrative database user of the MySQL server with privileges to shutdown the database. Symantec recommends creating a dedicated account in the database, with shutdown privileges only. Review the information for adding an administrative user for shutdown purposes. See "Adding a dedicated database administrator with shutdown privileges only" on page 18. Default Value: root Example: admin
MySQLAdminPasswd	Password for the database administrator specified in the MySQLAdmin attribute. The password is encrypted using the VCS encrypt utility, vcsencrypt(1m). **Note:** You need not encrypt the password if you are using the VCS GUI to enter the password. VCS GUI automatically encrypts the password. Default Value: "" Example : jxmXkvVvkVnvWvsVx
MySQLUser	The dedicated OS login created while installing the MySQL server. The database server will be started as this user. This login has to be identical on all failover nodes. Default Value: mysql Example: mysql
DataDir	The absolute path to the directory storing the database being managed by this instance of the server. Symantec recommends storing this directory on shared storage so that the same copy is available on the failover node. The database directory should be owned by the user specified by the MySQLUser agent attribute. Default Value: "" Example: /db/bbmas/data

Table 4-1 Required attributes *(continued)*

Attribute	Description
BaseDir	The installation path of the MySQL Database server. Default Value: "" Example: /usr/local/MySQL

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

Table 4-2 Optional attributes

Attribute	Description
EnvFile	Complete path of file name to source to set the environment prior to executing MySQL programs. Symantec recommends storing the file on the shared disk where the database directory (DataDir) is located. This ensures that the same file is available on each failover node. Specifying this attribute is optional. The shell environments supported are ksh, sh, and csh. Default Value: "" Example: /db/bbmas/envfile
HostName	Virtual host name for this MySQL Database instance. The monitor agent function uses this attribute to determine if the server is responding to client requests. This attribute is required only if second level monitoring is enabled. Default Value: "" Example: mysql.veritas.com
Port	Represents the port number dedicated to the MySQL server. The monitor agent function uses this value to determine if the server responds to client requests. This attribute is required only if second level monitoring is enabled. Default Value: 3306 Example: 3306

Table 4-2 Optional attributes *(continued)*

Attribute	Description
MonitorProgram	Absolute path name of an external, user-supplied monitor executable. For information about setting this attribute: See "Executing a customized monitoring program" on page 32. Default Value: "" Example 1.: /db/bbmas/myMonitor.pl Example 2.: /db/bbmas/myMonitor.sh arg1 arg2
SecondLevelMonitor	Used to enable second-level monitoring and specify how often it is run. Second-level monitoring is a deeper, more thorough state check of the configured MySQL instance. The numeric value specifies how often that the second-level monitoring routines are run. Care should be taken when setting this attribute 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 would only get performed every 100 minutes, which may not be as often as intended. To provide maximum flexibility, the value set is not checked for an upper limit. You can set the second level check to occur once a month, if that is desired. Default Value: 0 Example: 1
MyCnf	Complete path to the MySQL configuration file to be used while starting the database. Symantec recommends storing the file on the shared disk where the database directory (DataDir) is located. This ensures that the same file is available on each failover node. Default Value: "" Example: /db/bbmas/my.cnf

Executing a customized monitoring program

You can configure the monitor function to execute a custom monitor utility to perform a user-defined MySQL Server state check. The utility is executed in the context of the UNIX user that is defined in the MySQLUser attribute. The environment is set by sourcing the file specified in the EnvFile attribute.

The monitor operation executes MonitorProgram if:

- The MonitorProgram attribute value is set to a valid executable utility.
- The first level process check indicates that the MySQL Server instance is online.
- The SecondLevelMonitor attribute is set to 1 and the second level check returns the server state as ONLINE.
  Or
- The SecondLevelMonitor attribute is set to greater than 1, but the second level check is deferred for this monitoring cycle.

The monitor operation interprets the program exit code as follows:

110 or 0	MySQL server is online
100 or 1	MySQL server is offline
Any other value	MySQL server state is unknown

To ensure that the custom monitor utility is always available to the agent application, Symantec recommends storing the file in the directory in which the MySQL server gets installed.

Chapter 5

Configuring the service groups for MySQL

This chapter includes the following topics:

About configuring service groups for MySQL

Configuring the MySQL service group involves creating the MySQL 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

Before configuring the service groups for MySQL

Before you configure the MySQL 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 the Veritas agent for MySQL is installed on all nodes in the cluster.

See "Installing the agent in a VCS environment" on page 23.

MySQL entities in a clustered environment

A service group is a logical setup containing all resources that can support a MySQL instance in a clustered environment.

The required resources are as follows.

Disk group	Contains a volume and a file system, which is a mount resource containing the MySQL installation files. Use the DiskGroup resource type to create this resource. Create the disk group from the shared disk so that you can import the group into any system in the cluster.
Mount	Mounts, monitors, and unmounts the file system that is dedicated to the MySQL installation files. Use the Mount resource type to create this resource.
Network interface	Monitors the network interface card through which the MySQL instance communicates with other services. Use the NIC resource type to create this resource.
Virtual IP	Configures the virtual IP address dedicated to the MySQL instance. The external services, programs, and clients use this address to communicate with this instance. Use the IP resource type to create this resource.
MySQL server	Starts, stops, and monitors the MySQL instance. Use the MySQL resource type to create this resource.

Configuring MySQL resources for Solaris zones support

To enable the agent for MySQL to support Solaris zones, ensure that you perform the following configuration steps:

- Install MySQL on dedicated Solaris zones.
- Preferably, follow the Symantec recommendation of installing zones on a shared disk for convenient configuration, failover, and maintenance.
- Make sure that the name of the Solaris zone is the same as the virtual host name that you use to install and configure the MySQL.

- In a VCS environment, ensure that you have set the value of ContainerName attribute to the name of the Solaris zone.
  By default the agent function executes in the Global zone.

Chapter 6

Troubleshooting the agent for MySQL

This chapter includes the following topics:

Meeting prerequisites

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

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

See "Before you install the Veritas agent for MySQL" on page 21.

Verifying virtualization

Verify that your application does not use anything that ties it down to a particular node of the cluster.

See "Virtualizing MySQL " on page 18.

Starting the MySQL server outside 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 MySQL database server independent of the cluster framework. Refer to the cluster documentation for information about disabling a resource.

Note: Use the same parameters that the resource attributes defined within the cluster framework while restarting the resource outside the framework, like the owner of the application, the environment file etc.

- Starting the MySQL server
  To start the MySQL outside cluster, execute:

  ```
  $ BaseDir/bin/mysqld_safe --defaults-file=MyCnf \
   --datadir=DataDir --user=MySQLUser &
  ```

- Stopping the MySQL server
  To stop the MySQL outside cluster, execute:

  ```
  $ BaseDir/bin/mysqladmin --user=MySQLAdmin \
  --password=MySQLAdminPasswd shutdown
  ```

- Monitoring the MySQL server
  First verify if the MySQL processes are running as MySQLUser
- The agent uses a connect(3c) method to check for the MySQL server to listen to the port defined by the Port attribute. Try doing

  ```
  $ telnet HostName Port
  ```

  This should connect successfully.
- The agent then uses the following monitor command to verify that the MySQL server is up.

  ```
  $ BaseDir/bin/mysqladmin --user=MySQLAdmin --password=XXXXXX status
  Uptime: 2221700 Threads: 1 Questions: 35 Slow queries: 0 Opens:
  28 Flush tables: 1 Open tables: 4 Queries per second avg: 0.000
  $ echo $?
  0
  ```

  where *XXXXXX* is the password for the MySQLAdmin database user.
  The command is executed in the context of the MySQLUser. Try executing this command manually to verify if the MySQL server is up.

Reviewing error log files

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

The common reasons for issues are as follows:

Insufficient privileges	Files that need to be created or written to may be created as MySQLUser. Verify if necessary privileges have been set.
Incorrect port, environment or parameter settings	Verify that ports have been properly configured and declared. Typically, ports from 1 through 1024 are reserved for the superuser. Ensure that parameters to the agent are correctly defined.
Expired licenses	Check the application log files for any error messages related to expired licenses. Ensure that the license keys/files have been placed at the appropriate location, as needed by the application.
Broken symlinks, missing files, and libraries	Verify your installation. Make sure that nothing is broken, and all dependencies for the executables are met.
Insufficient disk space or system parameters	Ensure that the file-system has sufficient space for creation of temporary files that the application might need. Verify that the kernel has been tuned for sufficient IPC resources, file descriptors and meets the hardware requirement. Consult your product documentation for these details.

Consult your application expert if needed.

Using MySQL log files

MySQL by default writes error logs at *DataDir*/*HostName*.err, where *HostName* is the hostname of the node where the database is currently hosted.

Appendix A

Sample Configurations

This appendix includes the following topics:

About sample configurations for the agent for MySQL

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 MySQL. For more information about these resource types, see the *Veritas Cluster Server Bundled Agents Reference Guide*.

Sample agent type definition type for MySQL

VCS 4.x

```
type MySQL (
        static int ToleranceLimit = 1
        static str ArgList[] = { ResLogLevel, State,
        IState, MySQLUser, MySQLAdmin, MySQLAdminPasswd,
        EnvFile, BaseDir, DataDir, MyCnf, HostName,
        Port, SecondLevelMonitor, MonitorProgram }
        str ResLogLevel = INFO
        str MySQLUser = mysql
        str MySQLAdmin = root
        str MySQLAdminPasswd
```

```
        str EnvFile
        str BaseDir
        str DataDir
        str MyCnf
        str HostName
        int Port = 3306
        int SecondLevelMonitor = 0
        str MonitorProgram
)
```

VCS 5.0 (Solaris)

```
type MySQL (
        static int ToleranceLimit = 1
        static str ContainerType = Zone
        static boolean AEPTimeout = 1
        static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
        static str AgentDirectory = "/opt/VRTSagents/ha/bin/MySQL"
        static str ArgList[] = { ResLogLevel, State, IState,
        MySQLUser, MySQLAdmin, MySQLAdminPasswd, EnvFile,
        BaseDir, DataDir, MyCnf, HostName, Port, SecondLevelMonitor,
        MonitorProgram }
        str ResLogLevel = INFO
        str MySQLUser = mysql
        str MySQLAdmin = root
        str MySQLAdminPasswd
        str EnvFile
        str BaseDir
        str DataDir
        str MyCnf
        str HostName
        int Port = 3306
        int SecondLevelMonitor = 0
        str MonitorProgram
        str ContainerName
)
```

VCS 5.x (AIX, Linux, HPUX)

```
type MySQL (
        static int ToleranceLimit = 1
        static boolean AEPTimeout = 1
        static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
        static str AgentDirectory = "/opt/VRTSagents/ha/bin/MySQL"
```

```
        static str ArgList[] = { ResLogLevel, State, IState,
        MySQLUser,MySQLAdmin, MySQLAdminPasswd, EnvFile, BaseDir,
        DataDir, MyCnf, HostName, Port, SecondLevelMonitor,
        MonitorProgram }
        str ResLogLevel = INFO
        str MySQLUser = mysql
        str MySQLAdmin = root
        str MySQLAdminPasswd
        str EnvFile
        str BaseDir
        str DataDir
        str MyCnf
        str HostName
        int Port = 3306
        int SecondLevelMonitor = 0
        str MonitorProgram
)
```

VCS 5.1 (Solaris)

```
type MySQL (
        static int ToleranceLimit = 1
        static int ContainerOpts {} = { RunInContainer = 1,
        PassCInfo = 0 }
        static boolean AEPTimeout = 1
        static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
        static str AgentDirectory = "/opt/VRTSagents/ha/bin/MySQL"
        static str ArgList[] = { ResLogLevel, State, IState,
        MySQLUser, MySQLAdmin, MySQLAdminPasswd, EnvFile,
        BaseDir, DataDir, MyCnf, HostName, Port, SecondLevelMonitor,
        MonitorProgram }
        str ResLogLevel = INFO
        str MySQLUser = mysql
        str MySQLAdmin = root
        str MySQLAdminPasswd
        str EnvFile
        str BaseDir
        str DataDir
        str MyCnf
        str HostName
        int Port = 3306
        int SecondLevelMonitor = 0
        str MonitorProgram
```

Sample configuration files

A sample main.cf file for a configuration without zone support is as follows:

```
include "types.cf"
include "/etc/VRTSagents/ha/conf/MySQL/MySQLTypes50.cf"
cluster mysqls_cluster (
 UserNames = { root = ajkIjgJg }
 Administrators = { root }
 )
system Node1 (
 )
system Node2 (
 )
group SG_MySQL (
 SystemList = { Node1 = 0, Node2 = 1 }
 )
 DiskGroup RES_DiskGroup (
  DiskGroup = mysqldb
  )
 IP RES_Ip (
  Device = bge0
  Address = "10.209.60.194"
  NetMask = "0xfffffc00"
  )
 Mount RES_Mount (
  MountPoint = "/opt/mysql/mysql/shared_data   "
  BlockDevice = "/dev/vx/dsk/mysqldb/mysql_vol"
  FSType = vxfs
  FsckOpt = "-y"
  )
 MySQL RES_MySQL (
  MySQLAdmin = shutdown
  MySQLAdminPasswd = iwoUlwL
  BaseDir = "/opt/mysql/mysql"
  DataDir = "/opt/mysql/mysql/shared_data"
  MyCnf = "/etc/my.cnf"
  HostName = mysqlhost
  Port = 3307
  )
 NIC RES_Nic (
  Device = bge0
  NetworkHosts = { "10.209.60.1" }
```

```
 )
Volume RES_Volume (
 Volume = mysql_vol
 DiskGroup = mysqldb
 )
RES_Ip requires RES_Nic
RES_Mount requires RES_Volume
RES_MySQL requires RES_Ip
RES_MySQL requires RES_Mount
RES_Volume requires RES_DiskGroup

// resource dependency tree
//
// group SG_MySQL
// {
// MySQL RES_MySQL
//     {
//     Mount RES_Mount
//         {
//         Volume RES_Volume
//             {
//             DiskGroup RES_DiskGroup
//             }
//         }
//     IP RES_Ip
//         {
//         NIC RES_Nic
//         }
//     }
// }
```

A sample main.cf file for a configuration with zone support is as follows:

```
include "types.cf"
include "/etc/VRTSagents/ha/conf/MySQL/MySQLTypes50.cf"
cluster mysqls_cluster (
 UserNames = { root = ajkIjgJg,
   z_RESz_Zone_Node2 = eLKlLGlKKeMLjIJlMJ,
   z_RESz_Zone_Node1 = ajhEisGegGimHhkJim }
 Administrators = { root }
 )
system Node1 (
 )
```

```
system Node2 (
 )
group SGz_MySQL (
 SystemList = { Node1 = 0, Node2 = 1 }
 Administrators = { z_RESz_Zone_Node2, z_RESz_Zone_Node1 }
 )
 DiskGroup RESz_Dg (
  DiskGroup = mysql
  )
 Mount RESz_Mount (
  MountPoint = "/zones/mysql/"
  BlockDevice = "/dev/vx/dsk/mysql/mysql_vol_zone"
  FSType = vxfs
  FsckOpt = "-y"
  )
 MySQL RESz_MySQL (
  ResLogLevel = TRACE
  MySQLAdmin = mysql
  MySQLAdminPasswd = iwoUlwL
  BaseDir = "/opt/mysql/mysql/"
  DataDir = "/var/lib/mysql/"
  MyCnf = "/etc/my.cnf"
  HostName = mysql
  SecondLevelMonitor = 1
  ContainerName = mysql
  )
 NIC RESz_NIC (
  Device = bge0
  )
 Volume RESz_Vol (
  Volume = mysql_vol_zone
  DiskGroup = mysql
  )
 Zone RESz_Zone (
  ZoneName = mysql
  )
 RESz_Mount requires RESz_Vol
 RESz_MySQL requires RESz_Zone
 RESz_Vol requires RESz_Dg
 RESz_Zone requires RESz_Mount
 RESz_Zone requires RESz_NIC

 // resource dependency tree
```

```
//
// group SGz_MySQL
// {
// MySQL RESz_MySQL
//     {
//     Zone RESz_Zone
//         {
//         Mount RESz_Mount
//             {
//             Volume RESz_Vol
//                 {
//                 DiskGroup RESz_Dg
//                 }
//             }
//         NIC RESz_NIC
//         }
//     }
// }
```

Sample service group configurations for MySQL

This section includes sample service groups configurations in a VCS environment.

Figure A-1 shows a service group with a MySQL instance running in a VCS environment.

Figure A-1 Sample service group for a MySQL instance

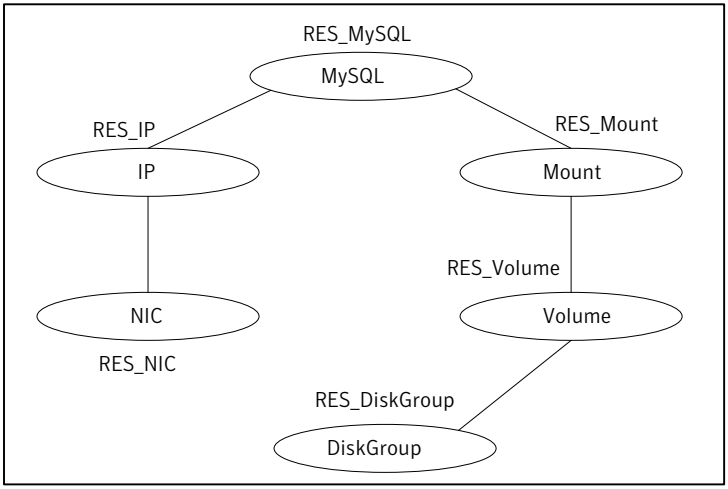
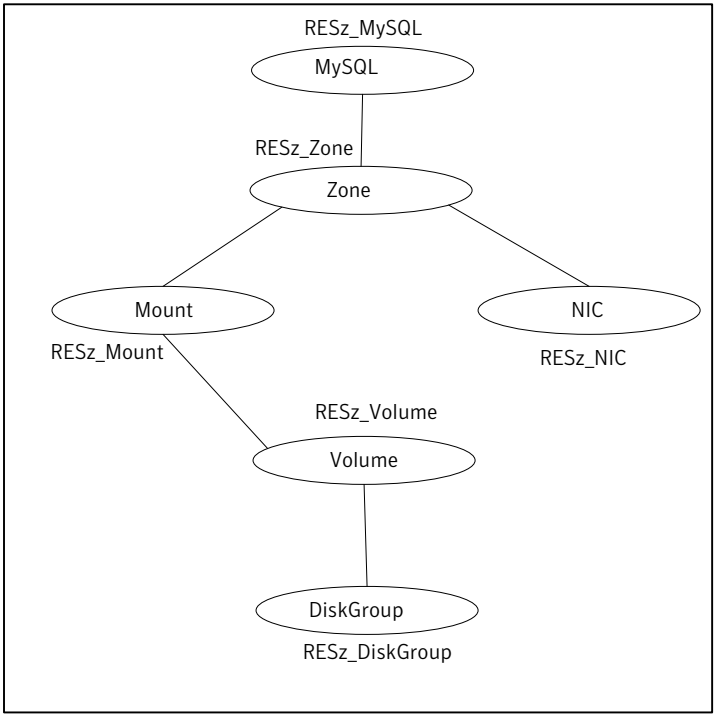


Figure A-2 shows a sample service group with Solaris zone support.

Figure A-2 Sample service groups with Solaris zone support



Index