

Cluster Server Agent for MySQL Installation and Configuration Guide

AIX, Linux, Solaris

8.0.2

Veritas InfoScale™ Availability Agents

Last updated: 2023-06-30

Legal Notice

Copyright © 2023 Veritas Technologies LLC. All rights reserved.

Veritas and the Veritas Logo are trademarks or registered trademarks of Veritas Technologies LLC or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

This product may contain third-party software for which Veritas is required to provide attribution to the third party ("Third Party Programs"). Some of the Third-Party Programs are available under open source or free software licenses. The License Agreement accompanying the Software does not alter any rights or obligations you may have under those open source or free software licenses. Refer to the third-party legal notices document accompanying this Veritas product or available at:

<https://www.veritas.com/about/legal/license-agreements>

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation/reverse engineering. No part of this document may be reproduced in any form by any means without prior written authorization of Veritas Technologies LLC and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. VERITAS TECHNOLOGIES LLC SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

The Licensed Software and Documentation are deemed to be commercial computer software as defined in FAR 12.212 and subject to restricted rights as defined in FAR Section 52.227-19 "Commercial Computer Software - Restricted Rights" and DFARS 227.7202, et seq. "Commercial Computer Software and Commercial Computer Software Documentation," as applicable, and any successor regulations, whether delivered by Veritas as on premises or hosted services. Any use, modification, reproduction release, performance, display or disclosure of the Licensed Software and Documentation by the U.S. Government shall be solely in accordance with the terms of this Agreement.

Veritas Technologies LLC
2625 Augustine Drive
Santa Clara, CA 95054

<http://www.veritas.com>

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:

<https://www.veritas.com/support>

You can manage your Veritas account information at the following URL:

<https://my.veritas.com>

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)

CustomerCare@veritas.com

Japan

CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The latest documentation is available on the Veritas website:

<https://sort.veritas.com/documents>

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:

infoscaledocs@veritas.com

You can also see documentation information or ask a question on the Veritas community site:

<http://www.veritas.com/community/>

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:

https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Chapter 1	Introducing the agent for MySQL	7
	About the Cluster Server agent for MySQL	7
	Features of the Cluster Server agent for MySQL	7
	Supported software	8
	How the agent makes MySQL highly available	8
	How the agent supports intelligent resource monitoring	9
	MySQL agent functions	9
	Online	9
	Offline	10
	Monitor	10
	Clean	12
	Setting up MySQL in a VCS cluster	12
 Chapter 2	 Installing and configuring MySQL for high availability	 13
	About MySQL	13
	Installing the MySQL instance	13
	Specifying shared disk for storing the database	14
	Setting MySQL parameters after installation	14
	Configuring the MySQL base directory and database directory	14
	Configuring virtual IP addresses	14
	Configuring port number and socket address	15
	Configuring the MySQL database user	15
	Adding a dedicated database administrator with shutdown privileges only	16
	Virtualizing MySQL	17
	Running multiple instances of MySQL on a single node	17
	About configuring MySQL for high availability	18
 Chapter 3	 Installing, upgrading, and removing the agent for MySQL	 19
	Before you install the Cluster Server agent for MySQL	19
	About the ACC library	20

	Installing the ACC library	20
	Installing the ACC library IPS package on Oracle Solaris 11 systems	21
	Installing the ACC library package on Solaris brand non-global zones	22
	Installing the agent in a VCS environment	23
	Installing the agent IPS package on Oracle Solaris 11 systems	24
	Installing agent packages on Solaris brand non-global zones	24
	Installing the agent in a Solaris 10 brand zone	25
	Uninstalling the agent in a VCS environment	26
	Removing the ACC library	27
	Upgrading the agent for MySQL	27
Chapter 4	Configuring the agent for MySQL	30
	About configuring the Cluster Server agent for MySQL	30
	Importing the agent types files in a VCS environment	30
	MySQL agent attributes	32
	Executing a customized monitoring program	38
Chapter 5	Enabling the agent for MySQL to support IMF	40
	About Intelligent Monitoring Framework	40
	Benefits of IMF	41
	Agent functions for the IMF functionality	41
	imf_init	41
	imf_getnotification	41
	imf_register	41
	Attributes that enable IMF	42
	IMF	42
	IMFRegList	43
	Before you enable the agent to support IMF	43
	Enabling the agent to support IMF	43
	If VCS is in a running state	44
	If VCS is not in a running state	46
	Disabling intelligent resource monitoring	46
	Sample IMF configurations	47

Chapter 6	Configuring the service groups for MySQL using the CLI	49
	Before configuring the service groups for MySQL	49
	MySQL entities in a clustered environment	49
	Configuring MySQL resources for Solaris zones support	50
Chapter 7	Troubleshooting the agent for MySQL	51
	Using the correct software and operating system versions	51
	Meeting prerequisites	51
	Verifying virtualization	52
	Starting the MySQL server outside a cluster	52
	Reviewing error log files	53
	Using MySQL log files	53
	Troubleshooting the configuration for IMF	54
	Known issues	55
Appendix A	Sample Configurations	56
	About sample configurations for the agents for MySQL	56
	Sample agent type definition type for MySQL	56
	Sample configuration files	59
	Sample service group configurations for MySQL	64

Introducing the agent for MySQL

This chapter includes the following topics:

- [About the Cluster Server agent for MySQL](#)
- [Features of the Cluster Server agent for MySQL](#)
- [Supported software](#)
- [How the agent makes MySQL highly available](#)
- [How the agent supports intelligent resource monitoring](#)
- [MySQL agent functions](#)
- [Setting up MySQL in a VCS cluster](#)

About the Cluster Server agent for MySQL

Cluster Server (VCS) agents monitor specific resources within an enterprise application. They determine the status of resources and start or stop them according to external events.

The Cluster Server agent for MySQL provides high availability for all the MySQL or MariaDB servers in a cluster.

For the latest updates or software issues for this agent, see the *Cluster Server Agent Pack Release Notes*.

Features of the Cluster Server 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

For information on the software versions that the Cluster Server agent for MySQL supports, see the Veritas Services and Operations Readiness Tools (SORT) site: <https://sort.veritas.com/agents>.

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 the hung or stopped states.
- **Secondary or Detail monitoring**
In this mode, the agent runs a utility to verify the status of the 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.
In addition to these levels of application monitoring, the agent for MySQL is IMF-aware and uses asynchronous monitoring framework (AMF) kernel driver for IMF notification.
Thus, the agent ensures high availability for MySQL instances.

How the agent supports intelligent resource monitoring

With Intelligent Monitoring Framework (IMF), VCS supports intelligent resource monitoring in addition to the poll-based monitoring. Poll-based monitoring polls the resources periodically whereas intelligent monitoring performs asynchronous monitoring.

When an IMF-enabled agent starts up, the agent initializes the Asynchronous Monitoring Framework (AMF) kernel driver. After the resource is in a steady state, the agent registers with the AMF kernel driver, the details of the resource that are required to monitor the resource.

For example, the agent for MySQL registers the PIDs of the MySQL processes with the AMF kernel driver.

The `imf_getnotification` function of the agent waits for any resource state changes. When the AMF kernel driver module notifies the `imf_getnotification` function about a resource state change, the agent framework runs the monitor agent function to ascertain the state of that resource. The agent notifies the state change to VCS, which then takes appropriate action.

For more information, see the *Cluster Server Administrator's Guide*.

MySQL agent functions

The operations or functions that the Cluster Server agent for MySQL can perform are 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
```

If the `mysqld_safe` script is not found, the agent uses the following command:

```
$ BaseDir/sbin/mysqld --defaults-file=MyCnf \
--datadir=DataDir --user=MySQLUser
```

The *BaseDir*, *MyCnf*, *DataDir*, *MySQLUser* variables in these commands represent the corresponding agent attribute values.

The commands always gets executed in the context of the user specified in the *MySQLUser* attribute. The default values to be used are specified in the MySQL configuration file, which is mentioned in the *MyCnf* attribute.

If SystemD is supported for the platform and if the *UseSystemD* attribute is set, the following command is used:

```
systemctl start serviceFileName
```

Note: On SystemD-enabled platforms, if more than one instance is running, the service name must be specified in the **mysql[d]@suffix** format.

- 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*.

If systemd is supported for the platform, the following command is used:

```
systemctl stop serviceFileName
```

Then the offline operation kills any existing processes that belong to this MySQL server instance, and 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.

The agent also supports Intelligent Monitoring Framework (IMF) in the first level check. IMF enables intelligent resource monitoring. The agent for MySQL is IMF-aware and uses the asynchronous monitoring framework (AMF) kernel driver for resource state change notifications. See [“How the agent supports intelligent resource monitoring”](#) on page 9.

You can use the MonitorFreq key of the IMF attribute to specify the frequency at which the agent invokes the monitor function. See [“MonitorFreq”](#) on page 42.

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 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.

If systemd is supported for the platform, the following command is used:

```
systemctl is-active serviceFileName
```

- Depending upon the MonitorProgram attribute, the monitor operation can perform a customized check using a user-supplied monitoring utility. For more information about the MonitorProgram attribute, See [“MySQL agent attributes”](#) on page 32.

Note: The attribute used to configure the second level check and its frequency depends on the software versions of VCS and MySQL agent you have installed: For VCS 5.1 SP1 or later with MySQL agent version 5.1.2.0, use the LevelTwoMonitorFreq attribute. For VCS 5.1 or earlier with MySQL agent 5.1.1.0 or earlier, use the SecondLevelMonitor attribute.

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.

If systemd is supported for the platform, the following command is used:

```
systemctl stop serviceName
```

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

Note: For information about the additional functions of the agent for MySQL when IMF is enabled: See [“Agent functions for the IMF functionality”](#) on page 41.

Setting up MySQL in a VCS cluster

Perform the following tasks to set up MySQL in a cluster:

1. Set up a VCS cluster.
2. Install and configure MySQL for high availability.
3. Install the Cluster Server agent for MySQL.
See [“Installing the agent in a VCS environment”](#) on page 23.
4. Configure the service groups for MySQL.

Installing and configuring MySQL for high availability

This chapter includes the following topics:

- [About MySQL](#)
- [Installing the MySQL instance](#)
- [Setting MySQL parameters after installation](#)
- [Adding a dedicated database administrator with shutdown privileges only](#)
- [Virtualizing MySQL](#)
- [About configuring MySQL for high availability](#)

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

See 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 32.

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 32.

Review the following sections while customizing the database configuration file:

- See [“Configuring the MySQL base directory and database directory”](#) on page 14.
- See [“Configuring virtual IP addresses”](#) on page 14.
- See [“Configuring port number and socket address”](#) on page 15.
- See [“Configuring the MySQL database user”](#) on page 15.

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 32.

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 `my.cnf` database configuration file through 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>

For more information about the HostName agent attribute,

See "MySQL agent attributes" on page 32.

Configuring port number and socket address

To ensure that multiple instances can be hosted on the same failover node, the port number and socket file address have to be unique. By default, the MySQL server listens on port 3306. The default values can be changed by updating the following values specified in the environment file by the EnvFile agent attribute:

- MYSQL_UNIX_PORT
- MYSQL_TCP_PORT

For more information about configuring the MYSQL_UNIX_PORT and MYSQL_TCP_PORT values, See "Running multiple instances of MySQL on a single node" on page 17.

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'@'127.0.0.1' ;
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 the MySQLAdmin and MySQLAdminPasswd attributes, See [“MySQL agent attributes”](#) on page 32.

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 through 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 15.

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.

About configuring MySQL for high availability

The guidelines for configuring MySQL for high availability are as follows:

- In a service group, keep the single point of failure as minimal as possible and watch the application startup time.
- Assign a virtual hostname to the component within the switchover environment. Because the physical hostname changes with the switchover, this is a must-have requirement.
- Based on the expected failover time, configure the reconnection parameters for all software components and enable its automatic reconnection.

Installing, upgrading, and removing the agent for MySQL

This chapter includes the following topics:

- [Before you install the Cluster Server agent for MySQL](#)
- [About the ACC library](#)
- [Installing the ACC library](#)
- [Installing the agent in a VCS environment](#)
- [Uninstalling the agent in a VCS environment](#)
- [Removing the ACC library](#)
- [Upgrading the agent for MySQL](#)

Before you install the Cluster Server agent for MySQL

You must install the Cluster Server agent for MySQL on all the systems that will host MySQL service groups.

Before you install the agent for MySQL, ensure that the following prerequisites are met.

- Install and configure Cluster Server.
For more information on installing and configuring Cluster Server, refer to the Cluster Server installation and configuration guides.

- Install the latest version of ACC Library.
To install or update the ACC Library package, locate the library and related documentation in the Agent Pack tarball.
See “[About the ACC library](#)” on page 20.

About the ACC library

The operations of a Cluster Server 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 ACCLib tar file has already been extracted.

Note: The LogDbg attribute should be used to enable debug logs for the ACCLib-based agents when the ACCLib version is 6.2.0.0 or later and VCS version is 6.2 or later.

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 a superuser.
- 2 Download ACC Library.

You can download either the complete Agent Pack tar file or the individual ACCLib tar file from the Veritas Services and Operations Readiness Tools (SORT) site (<https://sort.veritas.com/agents>).

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

AIX `cd1/aix/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 tar file, navigate to the pkgs directory (for AIX 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

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

Solaris      # pkgadd -d VRTSacclib.pkg
```

Note: The LogDbg attribute should be used to enable debug logs for the ACCLib-based agents when the ACCLib version is 6.2.0.0 or later and VCS version is 6.2 or later.

Installing the ACC library IPS package on Oracle Solaris 11 systems

Install the ACC library IPS package on an Oracle Solaris 11 system.

To install the ACC library IPS package on Oracle Solaris 11 systems

- 1 Copy the VRTSacclib.p5p package from the pkgs directory to the system in the /tmp/install directory.
- 2 Disable the publishers that are not reachable as package install may fail, if any, of the already added repositories are unreachable.

```
# pkg set-publisher --disable <publisher name>
```

- 3 Add a file-based repository in the system.

```
# pkg set-publisher -g /tmp/install/VRTSacclib.p5p Veritas
```

- 4 Install the package.

```
# pkg install --accept VRTSacclib
```

- 5 Remove the publisher from the system.

```
# pkg unset-publisher Veritas
```

- 6 Enable the publishers that were disabled earlier.

```
# pkg set-publisher --enable <publisher name>
```

Installing the ACC library package on Solaris brand non-global zones

With Oracle Solaris 11, you must install the agent package inside non-global zones. The native non-global zones are called Solaris brand zones.

To install the ACC library package on Solaris brand non-global zones

- 1 Ensure that the SMF services,
`svc:/application/pkg/system-repository:default` and
`svc:/application/pkg/zones-proxyd:default`, are online on the global zone.

```
# svcctl svc:/application/pkg/system-repository:default
# svcctl svc:/application/pkg/zones-proxyd:default
```
- 2 Log on to the non-global zone as a superuser.
- 3 Ensure that the SMF service
`svc:/application/pkg/zones-proxy-client:default` is online inside non-global zone:

```
# svcctl svc:/application/pkg/zones-proxy-client:default
```
- 4 Copy the `VRTSacclib.p5p` package from the `pkgs` directory to the non-global zone (for example, at the `/tmp/install` directory).
- 5 Disable the publishers that are not reachable, as package install may fail, if any of the already added repositories are unreachable.

```
# pkg set-publisher --disable <publisher name>
```
- 6 Add a file-based repository in the non-global zone.

```
# pkg set-publisher -g/tmp/install/VRTSacclib.p5p Veritas
```
- 7 Install the package.

```
# pkg install --accept VRTSacclib
```
- 8 Remove the publisher on the non-global zone.

```
# pkg unset-publisher Veritas
```
- 9 Clear the state of the SMF service, as setting the file-based repository causes the SMF service `svc:/application/pkg/system-repository:default` to go into the maintenance state.

```
# svcadm clear svc:/application/pkg/system-repository:default
```
- 10 Enable the publishers that were disabled earlier.

```
# pkg set-publisher --enable <publisher>
```

Note: Perform steps 2 through 10 on each non-global zone.

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 agent from the Veritas Services and Operations Readiness Tools (SORT) site: <https://sort.veritas.com/agents>.

You can download either the complete Agent Pack tar file or an individual agent tar file.

- 2 Uncompress the file to a temporary location, say /tmp.
- 3 If you downloaded the complete Agent Pack tar file, navigate to the directory containing the package for the platform running in your environment.

AIX	<code>cd1/aix/vcs/database/mysql_agent/ vcs_version/version_agent/pkg</code>
Linux	<code>cd1/linux/generic/vcs/database/mysql_agent/ vcs_version/version_agent/rpms</code>
Solaris	<code>cd1/solaris/dist_arch/vcs/database/mysql_agent/ vcs_version/version_agent/pkg</code> where, <i>dist_arch</i> is sol_sparc

If you downloaded the individual agent tar file, navigate to the pkgs directory (for AIX and Solaris), or rpms directory (for Linux).

- 4 Log in as a superuser.
- 5 Install the package.

AIX	<code># installp -ac -d VRTSmysql.rte.bff VRTSmysql.rte</code>
Linux	<code># rpm -ihv \ VRTSmysql-AgentVersion-GA_GENERIC.noarch.rpm</code>
Solaris	<code># pkgadd -d . VRTSmysql</code>

After installing the agent package, you must import the agent type configuration file.

See [“Importing the agent types files in a VCS environment”](#) on page 30.

Installing the agent IPS package on Oracle Solaris 11 systems

To install the agent IPS package on an Oracle Solaris 11 system

- 1 Copy the `VRTSmysql.p5p` package from the `pkgs` directory to the system in the `/tmp/install` directory.
- 2 Disable the publishers that are not reachable as package install may fail, if any of the already added repositories are unreachable.

```
# pkg set-publisher --disable <publisher name>
```

where the publisher name is obtained using the `pkg publisher` command.

- 3 Add a file-based repository in the system.

```
# pkg set-publisher -g /tmp/install/VRTSmysql.p5p Veritas
```

- 4 Install the package.

```
# pkg install --accept VRTSmysql
```

- 5 Remove the publisher from the system.

```
# pkg unset-publisher Veritas
```

- 6 Enable the publishers that were disabled earlier.

```
# pkg set-publisher --enable <publisher name>
```

Installing agent packages on Solaris brand non-global zones

To install the agent package on Solaris brand non-global zones

- 1 Ensure that the SMF services,
`svc:/application/pkg/system-repository:default` and
`svc:/application/pkg/zones-proxyd:default`, are online on the global zone.

```
# svcctl svc:/application/pkg/system-repository:default
```

```
# svcctl svc:/application/pkg/zones-proxyd:default
```

- 2 Log on to the non-global zone as a superuser.

3 Ensure that the SMF service

`svc:/application/pkg/zones-proxy-client:default` is online inside non-global zone:

```
# svcctl svc:/application/pkg/zones-proxy-client:default
```

4 Copy the `VRTSmysql.p5p` package from the `pkgs` directory to the non-global zone (for example, at the `/tmp/install` directory).**5** Disable the publishers that are not reachable, as package install may fail, if any of the already added repositories are unreachable.

```
# pkg set-publisher --disable <publisher name>
```

6 Add a file-based repository in the non-global zone.

```
# pkg set-publisher -g/tmp/install/VRTSmysql.p5p Veritas
```

7 Install the package.

```
# pkg install --accept VRTSmysql
```

8 Remove the publisher on the non-global zone.

```
# pkg unset-publisher Veritas
```

9 Clear the state of the SMF service, as setting the file-based repository causes the SMF service `svc:/application/pkg/system-repository:default` to go into the maintenance state.

```
# svcadm clear svc:/application/pkg/system-repository:default
```

10 Enable the publishers that were disabled earlier.

```
# pkg set-publisher --enable <publisher>
```

Note: Perform steps 2 through 10 on each non-global zone.

Installing the agent in a Solaris 10 brand zone

To install the MySQL agent in a Solaris 10 brand zone:

- Ensure that the ACC library package, `VRTSacclib`, is installed in the non-global zone.

To install `VRTSacclib` in the non-global zone, run the following command from the global zone:

```
# pkgadd -R /zones/zone1/root -d VRTSacclib.pkg
```

- To install the agent package in the non-global zone, run the following command from the global zone:

```
# pkgadd -R zone-root/root -d . VRTSmysql
```

For example: `# pkgadd -R /zones/zone1/root -d . VRTSmysql`

Note: You can ignore the following messages that might appear:

```
## Executing postinstall script.  
  
ln: cannot create /opt/VRTSagents/ha/bin/MySQL/imf_getnotification:  
File exists  
  
ln: cannot create /opt/VRTSagents/ha/bin/MySQL/imf_register: File  
exists  
  
or ## Executing postinstall script.  
  
ln: cannot create /opt/VRTSagents/ha/bin/MySQL/imf_getnotification:  
No such file or directory  
  
ln: cannot create /opt/VRTSagents/ha/bin/MySQL/imf_register: No such  
file or directory
```

Uninstalling 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 running the following command from any node in the cluster:


```
# haconf -makerw
```
- 3 Remove all MySQL resources from the cluster. Run 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 running 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 running 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.

Run the following command to uninstall the agent:

```
AIX          # installp -u VRTSmysql.rte
```

```
Linux        # rpm -e VRTSmysql
```

```
Solaris      # pkgrm VRTSmysql
```

Note: To uninstall the agent IPS package on a Solaris 11 system, run the following command:

```
# pkg uninstall 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
```

```
Linux        # rpm -e VRTSacclib
```

```
Solaris      # pkgrm VRTSacclib
```

Note: To uninstall the ACCLib IPS package on a Solaris 11 system, run the following command:

```
# pkg uninstall VRTSacclib
```

Upgrading the agent for MySQL

Perform the following steps to upgrade the agent with minimal disruption.

To upgrade the agent for MySQL

- 1 Verify the version of the agent for MySQL.

```
Linux # rpm -qi VRTSmysql | grep Version
```

```
Solaris # pkginfo -l VRTSmysql | grep VERSION
```

The output resembles:

```
Version : 7.0.0.0
```

- 2 Save the VCS configuration.

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

- 3 Identify the MySQL resource and service group.

```
# hatype -resources MySQL
```

The output resembles:

```
mysql
```

```
# hares -display mysql | grep Group
```

The output resembles:

```
mysql      Group      global      mysql_grp
```

- 4 Check the type-level attribute—Version.

```
# hatype -display MySQL | grep Version
```

- 5 Freeze the MySQL service group.

```
# hagrps -freeze mysql_grp
```

- 6 Check if the MySQL agent is running.

```
# haagent -display MySQL | grep Running
```

The output resembles:

```
MySQL      Running      Yes
```

- 7 If the agent is running, stop the agent.

```
# haagent -stop MySQL -force -sys hostname
```

8 Verify the status of the agent.

```
# haagent -display MySQL | grep Running
```

The output resembles:

```
MySQL          Running          No
```

9 Upgrade the agent.

- Linux # `rpm -Uvh VRTSmysql-agentVersion-GENERIC.noarch.rpm`

- For AIX and Solaris:

See [“Uninstalling the agent in a VCS environment”](#) on page 26.

See [“Installing the agent in a VCS environment”](#) on page 23.

10 Start the MySQL database.

```
# haagent -start MySQL -sys hostname
```

The output resembles:

```
VCS NOTICE V-16-1-10001 Please look for messages in the log file
```

11 Verify the status of the agent.

```
# haagent -display MySQL | grep Running
```

The output resembles:

```
MySQL          Running          Yes
```

12 Unfreeze the MySQL service group.

```
# hagr -unfreeze MySQL_grp
```

Configuring the agent for MySQL

This chapter includes the following topics:

- [About configuring the Cluster Server agent for MySQL](#)
- [Importing the agent types files in a VCS environment](#)
- [MySQL agent attributes](#)
- [Executing a customized monitoring program](#)

About configuring the Cluster Server agent for MySQL

After installing the Cluster Server agent for MySQL, you must import the agent type configuration file. After importing this file, review the attributes table that describes the resource type and its attributes, and then create and configure MySQL resources.

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

See [“About sample configurations for the agents for MySQL”](#) on page 56.

Importing the agent types files in a VCS environment

To use the agent for MySQL, you must import the agent types file into the cluster. You can import the agent types file using the VCS graphical user interface or using the command line interface.

To import the agent types file using the VCS Java GUI

- 1 Start the Cluster Manager (Java Console) 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 or later	■ AIX	/etc/VRTSagents/ha/conf/MySQL/
	■ Linux	MySQLTypes.cf
VCS 5.0	Solaris SPARC	/etc/VRTSagents/ha/conf/MySQL/ MySQLTypes50.cf
VCS 5.1 or later	Solaris SPARC	/etc/VRTSagents/ha/conf/MySQL/ MySQLTypes51.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 *Cluster Server Administrator's Guide*.

To import the agent types file using the CLI

- 1 If VCS is running, run the `/etc/VRTSagents/ha/conf/MySQL/MySQLTypes.cmd` file from the command line.
- 2 If VCS is not running, perform the following steps sequentially:

- Copy the agent types file from the `/etc/VRTSagents/ha/conf/<AgentTypes_file>` directory to the `/etc/VRTSvcs/conf/config` directory.
 Where, `<AgentTypes_file>` depends on the product version and the supported operating systems:

VCS 4.x	AIX	/etc/VRTSvcs/conf/sample_MySQL/
	Linux	MySQLTypes.cf
	Solaris	
VCS 5.x or later	AIX	/etc/VRTSagents/ha/conf/MySQL/
	Solaris	MySQLTypes.cf

VCS 5.0	Solaris SPARC	/etc/VRTSagents/ha/conf/MySQL/ MySQLTypes50.cf
VCS 5.1 or later	Solaris SPARC	/etc/VRTSagents/ha/conf/MySQL/ MySQLTypes51.cf

- Include the agent types file in the `main.cf` file.
- Start HAD.

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
MySQLAdmin	<p>The administrative database user of the MySQL server with privileges to shutdown the database. Veritas recommends creating a dedicated account in the database, with shutdown privileges only.</p> <p>Review the information for adding an administrative user for shutdown purposes.</p> <p>See “Adding a dedicated database administrator with shutdown privileges only” on page 16.</p> <p>Default Value: root</p> <p>Example: admin</p>
MySQLAdminPasswd	<p>The password for the database administrator specified in the MySQLAdmin attribute. The password is encrypted using the VCS encrypt utility, <code>vcseencrypt(1m)</code>.</p> <p>Note: You need not encrypt the password if you are using the VCS GUI to enter the password. The VCS GUI automatically encrypts the password.</p> <p>Default Value: No default value</p> <p>Example : <code>jxmXkvVvkVnvWvsVx</code></p>

Table 4-1 Required attributes (*continued*)

Attribute	Description
MySQLUser	<p>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.</p> <p>Default Value: mysql</p> <p>Example: mysql</p>
DataDir	<p>The absolute path to the directory storing the database being managed by this instance of the server. Veritas recommends storing this directory on shared storage so that the same copy is available on the failover node.</p> <p>The database directory should be owned by the user specified by the MySQLUser agent attribute.</p> <p>Default Value: No default value</p> <p>Example: /db/bbmas/data</p>
BaseDir	<p>The installation path of the MySQL Database server.</p> <p>Default Value: No default value</p> <p>Example: /usr/local/MySQL</p>

[Table 4-2](#) lists the optional attributes for the MySQL agent.

Table 4-2 Optional attributes

Attribute	Description
ResLogLevel	<p>Specifies the logging detail that the agent performs for the resource.</p> <p>The valid values are as follows:</p> <ul style="list-style-type: none"> ■ 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. <p>Default Value: INFO</p> <p>Example: INFO</p> <p>Note: You must use the LogDbg attribute instead of the ResLogLevel attribute to enable debug logs for the ACCLib-based agents, when the ACCLib version is 6.2.0.0 or later and the VCS version is 6.2 or later. The agent captures the first failure data of the unexpected events and automatically logs debug messages in their respective agent log files.</p>
LogDbg	<p>For ACCLib-based agents, you must use the LogDbg resource type attribute to enable the debug logs when the ACCLib version is 6.2.0.0 or later and the VCS version is 6.2 or later.</p> <p>Set the LogDbg attribute to DBG_5 to enable debug logs for ACCLIB based agent. By default, setting the LogDbg attribute to DBG_5 enables debug logs for all MySQL resources in the cluster. If debug logs must be enabled for a specific MySQL resource, override the LogDbg attribute.</p> <p>Type and dimension: string-keylist</p> <p>Default: {} (none)</p> <p>For more information on how to use the LogDbg attribute, refer to the <i>Cluster Server Administrator's Guide</i>.</p>

Table 4-2 Optional attributes (*continued*)

Attribute	Description
EnvFile	<p>The complete path of file name to source to set the environment prior to executing MySQL programs. Veritas 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.</p> <p>Default Value: No default value</p> <p>Example: /db/bbmas/envfile</p>
HostName	<p>The 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.</p> <p>This attribute is required only if second level monitoring is enabled.</p> <p>Default Value: No default value</p> <p>Example: mysql.veritas.com</p>
Port	<p>The port number dedicated to the MySQL server. This attribute is deprecated.</p> <p>Default Value: 3306</p> <p>Example: 3306</p>
MonitorProgram	<p>The absolute path name of an external, user-supplied monitor executable.</p> <p>For information about setting this attribute:</p> <p>See “Executing a customized monitoring program” on page 38.</p> <p>Default Value: No default value</p> <p>Example 1.: /db/bbmas/myMonitor.pl</p> <p>Example 2.: /db/bbmas/myMonitor.sh arg1 arg2</p>

Table 4-2 Optional attributes (*continued*)

Attribute	Description
SecondLevelMonitor	<p>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.</p> <p>Care should be taken when setting this attribute to large numbers.</p> <p>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.</p> <p>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.</p> <p>Note: The SecondLevelMonitor attribute is applicable to VCS versions earlier than VCS 5.1 SP1 with MySQL agent versions earlier than 5.1.2.0. From VCS version 5.1 SP1 or later with MySQL agent version 5.1.2.0 or later, the SecondLevelMonitor attribute of the MySQL agent is deprecated. Instead, a resource type level attribute LevelTwoMonitorFreq should be used to specify the frequency of in-depth monitoring.</p> <p>Default Value: 0</p> <p>Example: 1</p>
MyCnf	<p>The complete path to the MySQL configuration file to be used while starting the database. Veritas 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.</p> <p>Default Value: No default value</p> <p>Example: /db/bbmas/my.cnf</p>

Table 4-2 Optional attributes (*continued*)

Attribute	Description
LevelTwoMonitorFreq	<p>The frequency at which the agent for this resource type must perform second-level or detailed monitoring. You can also override the value of this attribute at the resource level.</p> <p>The value indicates the number of monitor cycles after which the agent will monitor the MySQL instance in detail. For example, the value 5 indicates that the agent will monitor the MySQL instance in detail after every five online monitor intervals.</p> <p>Note: This attribute is applicable to VCS version 5.1 SP1 or later with MySQL agent version 5.1.2.0 or later. If the VCS version is earlier than VCS 5.1 SP1 and the MySQL agent version is earlier than 5.1.2.0, use the SecondLevelMonitor attribute.</p> <p>If you upgraded the VCS version to VCS 5.1 SP1 or later and the MySQL agent version to 5.1.2.0 (or later), and if you had enabled detail monitoring in the previous version, then do the following:</p> <ul style="list-style-type: none">■ Set the value of the LevelTwoMonitorFreq attribute to the same value as that of the SecondLevelMonitor attribute. <p>Type and dimension: integer-scalar</p> <p>Default: 0</p>
ServiceName	<p>(SystemD support on RHEL 7 or later and SLES 12 or later only)</p> <p>The name of the unit service file that is used to start, stop, restart, or check the status of the MySQL or the MariaDB application. If this attribute is not specified, the agent uses the MySQL service name.</p> <p>Note: On SystemD-enabled platforms, if more than one instance is running, the service name must be specified in the mysql[d]@suffix format.</p> <p>Example: mysql or mysqld or mariadb</p> <p>Default: None</p> <p>Type and dimension: string-scalar</p>

Table 4-2 Optional attributes (*continued*)

Attribute	Description
MySQLPidFile	(SystemD support on RHEL 7 or later and SLES 12 or later only) The full path of the PID file, which stores the PID of the MySQL process. The location of the PID file is specified in the MySQL unit service file. Example: /var/run/mysqld/mysqld.pid Default: /var/run/mysqld/mysqld.pid Type and dimension: string-scalar
UseSystemD	SystemD is a system and a service manager for Linux operating systems. It helps manage applications across Linux distributions that support the SystemD feature. The UseSystemD attribute is applicable only on those systems that use the supported distributions of RHEL and SLES When this attribute is set to 1, the MySQL agent uses the MySQL service file for the online and the offline operations. The MySQL resource comes online as a service in system.slice. When this attribute is set to 0, a typical online function starts the resource in user.slice. Type and dimension: boolean-scalar Example: 1 Default: 0 Note: Set this attribute to 1 when the ServiceName attribute is set to mariadb .

Executing a customized monitoring program

You can configure the monitor function to execute MonitorProgram. MonitorProgram is 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 LevelTwoMonitorFreq attribute is set to 1 and the second-level check returns the server state as ONLINE.
 Or
- The LevelTwoMonitorFreq 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, Veritas recommends storing the file in the directory in which the MySQL server gets installed.

Enabling the agent for MySQL to support IMF

This chapter includes the following topics:

- [About Intelligent Monitoring Framework](#)
- [Agent functions for the IMF functionality](#)
- [Attributes that enable IMF](#)
- [Before you enable the agent to support IMF](#)
- [Enabling the agent to support IMF](#)
- [Disabling intelligent resource monitoring](#)
- [Sample IMF configurations](#)

About Intelligent Monitoring Framework

With the IMF feature, VCS supports intelligent resource monitoring in addition to the poll-based monitoring. Poll-based monitoring polls the resources periodically whereas intelligent monitoring performs asynchronous monitoring. You can enable or disable the intelligent resource monitoring functionality of the MySQL agent.

VCS process and mount-based agents use the AMF kernel driver that provides asynchronous event notifications to the agents that are enabled for IMF.

You can enable the MySQL agent for IMF, provided the following software versions are installed:

- Cluster Server (VCS) 5.1 SP1 or later
- Cluster Server agent for MySQL version 5.1.0.0 or later

Refer to the *Cluster Server Administrator's Guide* for more information about IMF notification module functions and administering the AMF kernel driver.

Benefits of IMF

IMF offers the following benefits:

- **Performance**
Enhances performance by reducing the monitoring of each resource at a default of 60 seconds for online resources, and 300 seconds for offline resources. IMF enables the agent to monitor a large number of resources with a minimal effect on performance.
- **Faster detection**
Asynchronous notifications would detect a change in the resource state as soon as it happens. Immediate notification enables the agent to take action at the time of the event.

Agent functions for the IMF functionality

If the MySQL agent is enabled for IMF support, the agent supports the following functions, in addition to the functions mentioned in the MySQL agent functions topic.

See [“MySQL agent functions”](#) on page 9.

imf_init

This function initializes the MySQL agent to interface with the AMF kernel driver, which is the IMF notification module for the agent for MySQL. This function runs when the agent starts up.

imf_getnotification

This function gets notifications about resource state changes. This function runs after the agent initializes with the AMF kernel module. This function continuously waits for notification and takes action on the resource upon notification.

imf_register

This function registers or unregisters resource entities with the AMF kernel module. This function runs for each resource after the resource goes into a steady state—online or offline.

Attributes that enable IMF

If the agent for MySQL is enabled for IMF support, the agent uses type-level attributes in addition to the agent-specific attributes.

See [“MySQL agent attributes”](#) on page 32.

IMF

This resource type-level attribute determines whether the MySQL agent must perform intelligent resource monitoring. You can also override the value of this attribute at the resource level.

This attribute includes the following keys:

Mode

Define this attribute to enable or disable intelligent resource monitoring.

The valid values are as follows:

- 0—Does not perform intelligent resource monitoring
- 1—Performs intelligent resource monitoring for offline resources and performs poll-based monitoring for online resources
- 2—Performs intelligent resource monitoring for online resources and performs poll-based monitoring for offline resources
- 3—Performs intelligent resource monitoring for both online and for offline resources.

Note: The agent supports intelligent resource monitoring for online resources only. Hence, Mode should be set to either 0 or 2.

Type and dimension: integer-association

Default: 0 for VCS 5.1 SP1, 3 for VCS 6.0 and later.

MonitorFreq

This key value specifies the frequency at which the agent invokes the monitor agent function. The value of this key is an integer.

Default: 1

You can set this key to a non-zero value for cases where the agent requires to perform both poll-based and intelligent resource monitoring.

If the value is 0, the agent does not perform poll-based process check monitoring.

After the resource registers with the AMF kernel driver, the agent calls the monitor agent function as follows:

- After every (MonitorFreq x MonitorInterval) number of seconds for online resources
- After every (MonitorFreq x OfflineMonitorInterval) number of seconds for offline resources

RegisterRetryLimit

If you enable intelligent resource monitoring, the agent invokes the `imf_register` agent function to register the resource with the AMF kernel driver.

The value of the `RegisterRetryLimit` key determines the number of times the agent must retry registration for a resource. If the agent cannot register the resource within the limit that is specified, then intelligent monitoring is disabled until the resource state changes or the value of the `Mode` key changes.

Default: 3.

IMFRegList

An ordered list of attributes whose values are registered with the IMF notification module.

Type and dimension: string-vector

Default: No default value

Note: The attribute values can be overridden at the resource level.

Before you enable the agent to support IMF

Before you enable the MySQL agent to support IMF, ensure that the AMF kernel module is loaded and AMF is configured. For details, refer to the 'Administering the AMF kernel driver' section of the *Cluster Server Administrator's Guide*. For details about the commands you can configure AMF using the `amfconfig -h` command.

Enabling the agent to support IMF

In order to enable the MySQL agent to support IMF, you must make the following configuration changes to the attributes of the agent:

- **AgentFile:** Set the AgentFile attribute to **Script51Agent** or **Script60Agent** as appropriate for your agent version
- **IMF Mode:** Set the IMF Mode attribute to **2**
- **IMFRegList:** Update the IMFRegList attribute

The following sections provide more information about the commands you can use to make these configuration changes, depending on whether VCS is in a running state or not.

Note: If you have upgraded VCS from an earlier version to version 5.1 SP1 or later, and you already have version 5.1.00 or later of the agent installed, ensure that you run the following commands to create appropriate symbolic links:

```
# cd /opt/VRTSagents/ha/bin/<resourceType>
# ln -s /opt/VRTSamf/imf/imf_getnotification imf_getnotification
# ln -s /opt/VRTSagents/ha/bin/<resourceType>/monitor imf_register
```

If VCS is in a running state

To enable the resource for IMF when VCS is in a running state

- 1 Make the VCS configuration writable.

```
# haconf -makerw
```

- 2 Run the following command to update the AgentFile attribute.

```
# hatype -modify <resourceType> AgentFile\
/opt/VRTSvc/bin/Script51Agent
```

- 3 For VCS version 6.0 or later, run the following commands to add the IMF attributes:

```
# haattr -add -static <resourceType> IMF -integer -assoc Mode 0 \
MonitorFreq 1 RegisterRetryLimit 3
```

```
# haattr -add -static <resourceType> IMFRegList -string -vector
```

Note: Run these commands only once after you first enable IMF support for the agent.

- 4 Run the following command to update the IMF attribute.

```
# hatype -modify <resourceType> IMF Mode num MonitorFreq num  
RegisterRetryLimit num
```

For example, to enable intelligent monitoring of online resources, with the MonitorFreq key set to 5, and the RegisterRetryLimit key is set to 3, run the following command:

```
# hatype -modify <resourceType> IMF Mode 2 MonitorFreq 5 \  
RegisterRetryLimit 3
```

Note: The valid values for the Mode key of the IMF attribute are 0 (disabled) and 2 (online monitoring).

- 5 Run the following command to update the IMFRegList attribute:

```
# hatype -modify MySQL IMFRegList BaseDir DataDir MySQLUser
```

- 6 Save the VCS configuration.

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

- 7 If the agent is running, restart the agent.

For information on the commands you can use to restart the agent, see [Restarting the agent](#).

Restarting the agent

To restart the agent:

- 1 Run the following command to stop the agent forcefully:

```
# haagent -stop <resourceType> -force -sys <systemName>
```

Note: Stopping the agent forcefully eliminates the need to take the resource offline.

- 2 Run the following command to start the agent:

```
# haagent -start <resourceType> -sys <systemName>
```

If VCS is not in a running state

To change the agent type definition file when VCS is not in a running state

- 1 Update the AgentFile attribute.

```
static str AgentFile = "/opt/VRTSvcs/bin/Script51Agent"
```

- 2 Update the IMF attribute.

The valid values for the Mode key of the IMF attribute are 0 (disabled) and 2 (online monitoring).

```
static int IMF{} = { Mode=num, MonitorFreq=num,  
RegisterRetryLimit=num }
```

For example, to update the IMF attribute such that the Mode key is set to 2, the MonitorFreq key is set to 5, and the RegisterRetryLimit key is set to 3:

```
static int IMF{} = { Mode=2, MonitorFreq=5, RegisterRetryLimit=3  
}
```

- 3 Update the IMFRegList attribute.

```
static str IMFRegList[] = { BaseDir, DataDir, MySQLUser }
```

Disabling intelligent resource monitoring

To disable intelligent resource monitoring

- 1 Make the VCS configuration writable.

```
# haconf -makerw
```

- 2 To disable intelligent resource monitoring for all the resources of a certain type, run the following command:

```
# hatype -modify MySQL IMF -update Mode 0
```

- 3 To disable intelligent resource monitoring for a specific resource, run the following command:

```
# hares -override resource_name IMF
```

```
# hares -modify resource_name IMF -update Mode 0
```

- 4 Save the VCS configuration.

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

Sample IMF configurations

An example of a type definition file for a MySQL agent that is IMF-enabled is as follows:

In this example, the IMF-related attributes are set to the following values:

AgentFile	/opt/VRTSvcs/bin/Script51Agent
IMF{}	{ Mode=2, MonitorFreq=5, RegisterRetryLimit=3 }
IMFRegList[]	{ BaseDir DataDir MySQLUser }
LevelTwoMonitorFreq	25

```
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
    static int IMF{} = { Mode=2, MonitorFreq=5, RegisterRetryLimit=3 }
    static str IMFRegList[] = { BaseDir DataDir MySQLUser }
)
```

A sample resource configuration from the `/etc/VRTSvcs/conf/config/main.cf` file is as follows:

```
MySQL mysql (
    Critical = 0
```

```
MySQLAdminPasswd = iwlWruVujUwwMunUl  
BaseDir = "/opt/mysql/mysql"  
DataDir = "/var/lib/mysql"  
HostName = "vcssx074.vxindia.veritas.com"  
)
```


Configuring the service groups for MySQL using the CLI

This chapter includes the following topics:

- [Before configuring the service groups for MySQL](#)
- [MySQL entities in a clustered environment](#)
- [Configuring MySQL resources for Solaris zones support](#)

Before configuring the service groups for MySQL

Before you configure the MySQL service group, you must:

- Verify that the Cluster Server components are installed and configured on all nodes in the cluster where you will configure the service group.
For more information on installing the components, refer to the *InfoScale Availability Installation Guide*.
- Verify that the Cluster Server 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	<p>Contains a volume and a file system, which is a mount resource containing the MySQL installation files.</p> <p>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.</p>
Mount	<p>Mounts, monitors, and unmounts the file system that is dedicated to the MySQL installation files.</p> <p>Use the Mount resource type to create this resource.</p>
Network interface	<p>Monitors the network interface card through which the MySQL instance communicates with other services.</p> <p>Use the NIC resource type to create this resource.</p>
Virtual IP	<p>Configures the virtual IP address dedicated to the MySQL instance. The external services, programs, and clients use this address to communicate with this instance.</p> <p>Use the IP resource type to create this resource.</p>
MySQL server	<p>Starts, stops, and monitors the MySQL server instance.</p> <p>Use the MySQL server resource type to create this resource.</p>

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 Veritas 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.

Troubleshooting the agent for MySQL

This chapter includes the following topics:

- [Using the correct software and operating system versions](#)
- [Meeting prerequisites](#)
- [Verifying virtualization](#)
- [Starting the MySQL server outside a cluster](#)
- [Reviewing error log files](#)
- [Troubleshooting the configuration for IMF](#)

Using the correct software and operating system versions

Ensure that you use correct software and operating system versions.

For information on the software versions that the agent for MySQL supports, see the Veritas Services and Operations Readiness Tools (SORT) site:

<https://sort.veritas.com/agents>.

Meeting prerequisites

Before installing the agent for MySQL, ensure that the following prerequisites are met.

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

See [“Before you install the Cluster Server agent for MySQL”](#) on page 19.

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 17.

Starting the MySQL server 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 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 server outside cluster, execute:

```
$ baseDirectory/bin/mysqld_safe --defaults-file=MyCnf \
  --datadir=dataDirectory --user=mysqlUser &
```

- **Stopping the MySQL server**

To stop the MySQL server outside cluster, execute:

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

- **Monitoring the MySQL server**

First verify that the MySQL processes are running as MySQLUser.

- **The agent uses the following monitor command to verify that the MySQL server is up.**

```
$ baseDirectory/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.

Troubleshooting the configuration for IMF

If you face problems with the IMF configuration or functionality, consider the following:

- Ensure that the following attributes are configured with appropriate values.
 - AgentFile
 - IMF
 - IMFRegList

If IMFRegList is not configured correctly, the MySQL resources that have been registered for IMF get unregistered every time the monitor function is run.
- If you have configured the required attributes to enable the MySQL agent for IMF, but the agent is still not IMF-enabled, restart the agent. The `imf_init` function runs only when the agent starts up, so when you restart the agent, `imf_init` runs and initializes the MySQL agent to interface with the AMF kernel driver.

- You can run the following command to check the value of the `MonitorMethod` attribute and to verify that a resource is registered for IMF.

```
# hares -value resource MonitorMethod system
```

The `MonitorMethod` attribute specifies the monitoring method that the agent uses to monitor the resource:

- Traditional—Poll-based resource monitoring
- IMF—Intelligent resource monitoring
- You can use the `amfstat` to see a list of registered PIDs for a MySQL resource. A sample output of the `ps -ef` command for the MySQL processes is as follows:

```
$/usr/ucb/ps auxwwl | grep mysql
0  551  1646      1  0  59 20 1688 1304 6001cb5b198 S ?
0:00 sh -c /opt/mysql/mysql/bin/mysqld_safe
--datadir=/var/lib/mysql --user=mysql
0  551  1648  1646  0  59 20 1744 1360 6001caec108 S ?
0:00 /bin/sh /opt/mysql/mysql/bin/mysqld_safe
--datadir=/var/lib/mysql --user=mysql
0  551  1695  1648  0  59 206660834160 6001c997682 S ?
0:00 /opt/mysql/mysql/bin/mysqld
--basedir=/opt/mysql/mysql --datadir=/var/lib/mysql
--user=mysql --log-error=/var/lib/mysql/vcssx074.vxindia.
veritas.com.err --pid-file= /var/lib/mysql/vcssx074.
vxindia.veritas.com.pid 0      0  1747   550  0  49 20 1640
```

```
1152 6001ce02a9c S pts/1 0:00 grep mysql
```

The `amfstat` command shows the PIDs monitored by the agent.

```
Registered Reapers (1):
=====
RID      PID      EVENT    REAPER
1        568      3         0         MySQL

Process ONLINE Monitors (3):
=====
RID      R_RID    PID      GROUP
22       1        1695     mysql
23       1        1646     mysql
24       1        1648     mysql
```

- Run the following command to set the `ResLogLevel` attribute to `TRACE`. When you set `ResLogLevel` to `TRACE`, the agent logs messages in the `MySQL_A.log` file.

```
# hares -modify ResourceName ResLogLevel TRACE
```

For more information about the `ResLogLevel` attribute, See [“MySQL agent attributes”](#) on page 32.

- Run the following command to view the content of the AMF in-memory trace buffer.

```
# amfconfig -p dbglog
```

Known issues

This release of the agent for MySQL has the following known issues:

Problem

An error message might appear when you run the `hares -offline` command to take a resource offline.

Description

When a resource is taken offline, it is unregistered from the AMF module. However, the `imf_register` function attempts to unregister the resource again.

This results in an error message from the engine log.

Workaround

It is safe to ignore this error message.

Sample Configurations

This appendix includes the following topics:

- [About sample configurations for the agents for MySQL](#)
- [Sample agent type definition type for MySQL](#)
- [Sample configuration files](#)
- [Sample service group configurations for MySQL](#)

About sample configurations for the agents 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 agents for MySQL. For more information about these resource types, refer to the *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
)
```

(Solaris only) VCS 5.0

```
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, ServiceName, MySQLPidFile,
SecondLevelMonitor, MonitorProgram, UseSystemD }
    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
    str ServiceName
    str MySQLPidFile = "/var/run/mysqld/mysqld.pid"
    boolean UseSystemD = 1
)
```

(AIX and Linux only) VCS 5.x

```
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, ServiceName, MySQLPidFile,  
SecondLevelMonitor, MonitorProgram, UseSystemD }  
    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 ServiceName  
    str MySQLPidFile = "/var/run/mysqld/mysqld.pid"  
    boolean UseSystemD = 1  
)
```

(Solaris only) VCS 5.1

```
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, ServiceName, MySQLPidFile,  
SecondLevelMonitor, MonitorProgram, UseSystemD }  
    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 ServiceName
    str MySQLPidFile = "/var/run/mysqld/mysqld.pid"
    boolean UseSystemD = 1
)
```

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 = mysqlldb
)
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/mysqlldb/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 = mysqlldb
)
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
//         }
//     }
// }

```

A sample `main.cf` file for MariaDB configuration on Linux:

```

include "types.cf"
include "/etc/VRTSagents/ha/conf/MySQL/MySQLTypes.cf"
cluster mariadb_cluster (
    UserNames = { root = ajkIjgJg }
    Administrators = { root }
)
system Node1 (
)
system Node2 (

```

```
)
group SG_MariaDB (
SystemList = { Node1 = 0, Node2 = 1 }
)
DiskGroup RES_DiskGroup (
DiskGroup = dgmaria
)
IP RES_Ip (
Device = bge0
Address = "10.209.60.194"
NetMask = "0xfffffc00"
)
Mount RES_Mount (
MountPoint = "/opt/mariadb/mariadb/shared_data "
BlockDevice = "/dev/vx/dsk/dgmaria/maria_vol"
FSType = vxfs
FsckOpt = "-y"
)

MySQL RES_MariaDB (
MySQLAdminPasswd = iwoUlwL
BaseDir = "/opt/mariadb/mariadb"
DataDir = "/opt/mariadb/mariadb/shared_data"
MyCnf = "/etc/my.cnf"
HostName = mariadbHost
Port = 3307
ServiceName = mariadb
)
NIC RES_Nic (
Device = bge0
NetworkHosts = { "10.209.60.1" }
)
Volume RES_Volume (
Volume = maria_vol
DiskGroup = dgmaria
)
RES_Ip requires RES_Nic
RES_Mount requires RES_Volume
RES_MariaDB requires RES_Ip
RES_MariaDB requires RES_Mount
RES_Volume requires RES_DiskGroup

// resource dependency tree
```

```
//
// group SG_MariaDB
// {
//   MySQL RES_MariaDB
//   {
//     Mount RES_Mount
//     {
//       Volume RES_Volume
//       {
//         DiskGroup RES_DiskGroup
//       }
//     }
//     IP RES_Ip
//     {
//       NIC RES_Nic
//     }
//   }
// }
```

Sample service group configurations for MySQL

Figure A-1 Sample service group for a MySQL instance

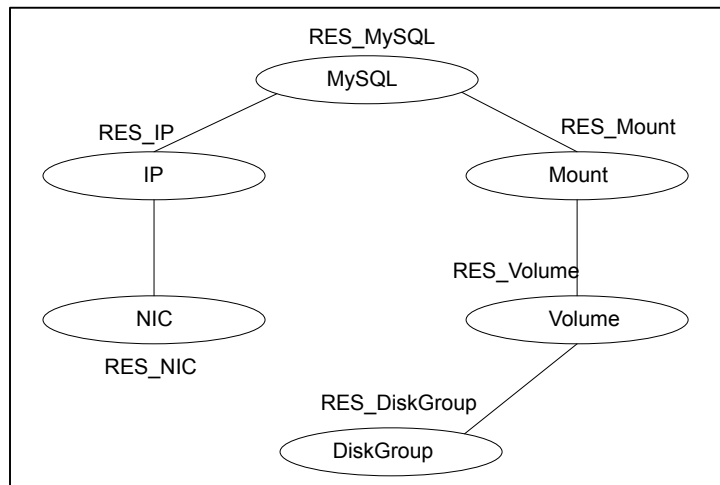


Figure A-2 Sample service group for a MariaDB instance

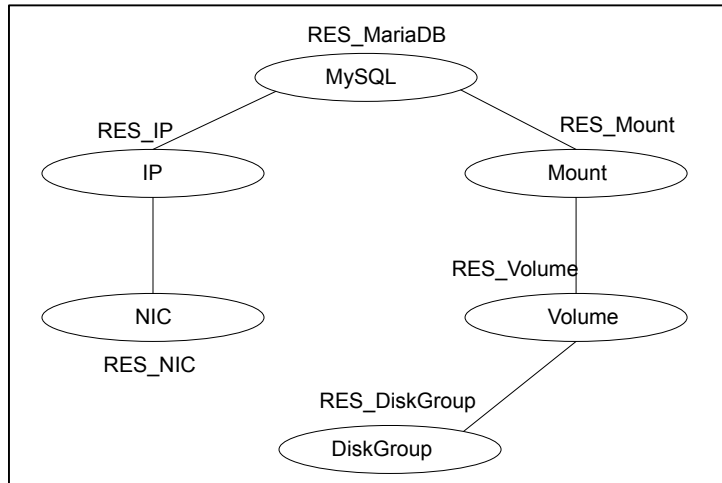


Figure A-3 Sample service group configurations with Solaris zone support

