SIOS Protection Suite for Linux

MySQL Recovery Kit

v9.3.1

Administration Guide

November 2018

Table of Contents

Chapter 1: Introduction

MySQL Recovery Kit Documentation

The SIOS Protection Suite for Linux MySQL Recovery Kit provides an easy way to add LifeKeeper fault-resilient protection for MySQL resources and databases. This enables a failure on the primary database server to be recovered on a designated backup server without significant lost time or human intervention.

SIOS Protection Suite Documentation

The following SPS product documentation is available from SIOS Technology Corp.:

- SPS for Linux Release Notes
- SPS for Linux Technical Documentation (also available from the **Help** menu within the LifeKeeper GUI)

This documentation, along with documentation associated with optional LifeKeeper Recovery Kits, is available on the SIOS Technical Documentation website.

Requirements

Kit Hardware and Software Requirements

Before you can install and set up the recovery software, your server must meet certain hardware and software requirements. You should refer to the SPS for Linux Installation Guide for specific instructions on how to install or remove the LifeKeeper MySQL Recovery Kit.

Be sure that your configuration meets the following requirements:

- **Servers**. The Recovery Kit requires two or more LifeKeeper supported computers configured in accordance with the requirements described in SPS for Linux Technical Documentation and the SPS for Linux Release Notes.
- **LifeKeeper software**. You must install the same version of LifeKeeper software and any patches on each server. Please refer to the SPS for Linux Technical Documentation and the SPS for Linux Release Notes for specific LifeKeeper requirements.
- **LifeKeeper IP Recovery Kit**. This kit is required if remote clients will be accessing the MySQL database. You must have the same version of this Recovery Kit on each server.
- **IP network interface**. Each server requires at least one Ethernet TCP/IP-supported network interface. In order for IP switchover to work properly, user systems connected to the local network should conform to standard TCP/IP specifications.

Note: Even though each server requires only a single network interface, you should use multiple interfaces for a number of reasons: heterogeneous media requirements, throughput requirements, elimination of single points of failure, network segmentation and so forth.

- **TCP/IP software**. Each server also requires the TCP/IP software.
- **MySQL software**. Each server must have the MySQL software installed and configured prior to configuring LifeKeeper and the LifeKeeper MySQL Recovery Kit. The same version should be installed on each server. Consult the SPS for Linux Release Notes or your sales representative for the latest release compatibility and ordering information.

Chapter 2: Configuration Considerations

This section contains definitions and examples of typical LifeKeeper MySQL configurations and information you should consider before you start to configure MySQL.

Please refer to the Resource Hierarchies section of the SPS for Linux Technical Documentation for instructions on configuring your LifeKeeper Core resource hierarchies.

Configuration Considerations for MySQL

Below are some specific considerations you need to think about concerning your LifeKeeper MySQL environment.

To operate MySQL database services on the primary and backup servers, file systems and disk partitions must be accessible from each server. Before you can begin configuring the MySQL Recovery Kit, be sure you have completed the following preliminary steps and have tested/run the databases on each server. In the instructions below, the user “mysql” refers to the operating system user that will start the MySQL server.

1. Install the MySQL server and client components on all servers. Be sure that all of the servers are running the same version of the MySQL client and server components. The MySQL executables can be located on a local or shared drive.

   Note:If you use Red Hat Software Collections and need to export the X_SCLS environment variable in order to run a specific version of MySQL with LifeKeeper, then set the X_SCLS environment variable via `/etc/default/LifeKeeper` by adding the line X_SCLS=VERSION to the file (i.e. X_SCLS=mysql55). This is typically only the case if you want to enable MySQL 5.5 which is included in RHEL 5.10 (MySQL 5.0 is enabled as the default).

2. If mysqld is running on any of the servers on the socket and/or port where you wish to run the LifeKeeper protected MySQL database server, stop each MySQL server using the `mysqladmin` command.

3. Move the contents of the MySQL data directory to a shared location. By default, the MySQL data directory is installed on a local drive. This location depends on the distribution mechanism. The binary RPM installs the data directory at */var/lib/mysql*. (Be sure that only the contents are moved and the directory remains intact. This allows the MySQL database server to write logs in this directory, if necessary. Make sure that the “mysql” user described in step 4 has permissions to write the logs to this location.)

4. If the installation process did not create the Linux user “mysql”, create this user. For security reasons, the MySQL server should not be run as “root.” (Refer to the *MySQL Administration Guide* for a full discussion of the security issues.) Make sure that “mysql” is the only user with read/write permissions in the database directories. The “mysql” user and group should be created on all servers. The user ID and group ID must be the same on all servers.

5. **IMPORTANT**: A server started by /etc/rc.d/init.d/mysql cannot be under LifeKeeper protection. In addition, the server can not use the same port number or socket as a server under LifeKeeper protection.

6. It is recommended that the socket be written to the data directory on the shared disk. If the socket will be written to a local disk, make sure the path exists on all LifeKeeper servers where your hierarchy will exist. Make sure that the user “mysql” has permissions to write the socket to this location.

7. Start the MySQL server using the mysql daemon startup command appropriate for your configuration. For configurations defining a single instance in the *my.cnf* file, use the command:

   ```
   <start command> --user=mysql --socket=<socket> --port=<port number>

   --datadir=<path to the data directory> --log &
   ```

   The <start command> for mysql versions 3.x is `safe_mysqld`, and the command for version 4.x is `mysqld_safe`.

   For configurations using the mysqld Group feature in the *my.cnf* file, use the command:

   ```
   mysqld_multi start <group number>
   ```

   The <group number> represents the numerical instance defined in the *my.cnf* file for the mysqld Group. For more information on using mysqld groups with LifeKeeper, see: Using mysqld Groups with LifeKeeper.

   sytemctl command must be utilized when MySQL (v5.7.6 or later) is set up to use Systemd in the distribution with Systemd. For the details, refer to "Consideration about the use in Systemd environment".

8. Create a MySQL database user named “mysql”. Give this user a password and grant the user “shutdown” permissions. This only has to be done on one server. (Refer to the *MySQL Administration Guide* for details on creating users and granting permissions).

9. Copy the sample my.cnf configuration file to the desired location (*/etc* or */<datadir>*). This file contains options for the database server and for client programs.

   The file can be located in either the *MySQL data* directory or the */etc* directory. The */etc/my.cnf* file contains global options. Place the *my.cnf* file in */etc* if only one database will run on the machine at any given time (i.e. an Active/Standby configuration) or if you are using the mysqld Group feature (see Using mysqld Groups with LifeKeeper). If the file is located in */etc*, you must copy it to each LifeKeeper backup server. The *my.cnf* file in the data directory should contain server-specific options. For multiple servers and Active/Active configurations, this file must be stored in the data directory for each resource instance unless you are using the mysqld Group feature (see Using mysqld Groups with LifeKeeper).

   Note: The *my.cnf* file should not exist in both the */etc* and */<datadir>* locations if both copies will contain server specific options. If a *my.cnf* file containing server specific options is located in */etc* along with a protected *my.cnf* file installed in the */<datadir>* potential conflicts may result. Refer to the MySQL documentation on configuring global settings and server specific options.

   Add or edit the following entries:

a. In the “client” section of the file, specify the user and the password that should be used for connections.

```
[client]
 user =clientuser
 password =password
 .
 .
 .
```

b. In the “mysqld” section of the file, specify the socket and port that should be used for connections, as well as the pid-file location for the mysqld process. The user variable should specify the operating system user that will start the mysqld process.

```
[mysqld]
 socket =/home1/test/mysql/mysql.sock
 port =3307
 pid-file =/home1/test/mysql/mysqld.pid
 user =osuser
```

Note: Make sure this file is properly protected and owned by the user “mysql.”

Note: Once the MySQL hierarchy is created, if you need to change any of the information in the *my.cnf* file, you must stop the mysql server instance by taking the hierarchy out-of-service (i.e. the OSU state) before making changes.

Note: The above example *my.cnf* configuration describes a single database instance *mysqld*. See Using mysqld Groups with LifeKeeper for configuration examples using mysqld groups.

Note: "include" directive is not supported. All the setups must be described in a single my.cnf file.

Client Configuration Considerations

Following are some configuration considerations for MySQL database clients:

- If clients will connect from remote hosts, create an IP address under LifeKeeper to be used for client connections.
- Clients must be configured to connect to the database server through a LifeKeeper-protected IP address.
- If the clients will connect through a domain name instead, create an entry in each client’s hosts file for the protected IP address, or configure the name in DNS. Test the protected IP address by pinging it from all clients and all LifeKeeper servers in the cluster.
- Although each user can have a *my.cnf* file in the home directory of their machine, LifeKeeper only uses the *my.cnf* file located in the */etc* directory or the data directory. The *my.cnf* file stores the client connection information (i.e. the port, socket identification, user and password).

Configuration Requirements

Each of the examples involves one or two database instances: databaseA and databaseB. The Database Tag names are arbitrary names that describe these database instances to LifeKeeper. The word on and the system identifier that follows provide clarification but are not required. The default tag name suggested by LifeKeeper is mysql or mysql<group number> for configurations using mysqld Groups (see Using mysqld Groups with LifeKeeper). To understand the configuration examples, keep these configuration requirements in mind:

- **LifeKeeper hierarchy**. When performing LifeKeeper administration, the primary hierarchy refers to the hierarchy being built on the server you are administering. For the configuration diagrams, the information entered in the first administration screen is from the perspective of Server 1. When a second screen is shown, it refers to the hierarchy being built while administering the second server. In the configuration examples, the second server is Server 2.
- **Shared disk locked by one server**. When you use LifeKeeper, one server reserves shared storage resources that are under LifeKeeper protection for use. This is done using SCSI reservations. If the shared device is a disk array, an entire LUN is reserved; if a shared device is a disk, then the entire disk is reserved. This prevents inadvertent corruption of the data by other servers in the cluster. When a server fails, the highest priority backup server breaks the old reservation and establishes its own reservation, locking out all other servers.
- **Data Directory on shared disk**. In order for the LifeKeeper MySQL Recovery Kit to function properly, the data directory (datadir) of the database instance must always be on a shared disk. The data directory must be on a file system. The file system must be mountable from both the primary and backup servers. The data directory (datadir) can also exist on replicated or network attached storage.

Configuration Examples

The examples in this section show how MySQL database instances can be configured. Each diagram shows the relationship between the type of configuration and the MySQL parameters. Each configuration also adheres to the configuration rules and requirements described in this documentation that ensure compatibility between the MySQL configuration and the LifeKeeper software.

This section describes the configuration requirements and then provides these configuration examples:

- Active/Standby
- Active/Active

The examples in this section are only a sample of the configurations you can establish, but understanding these configurations and adhering to the configuration rules will help you define and set up workable solutions for your computing environment.

Configuration Requirements

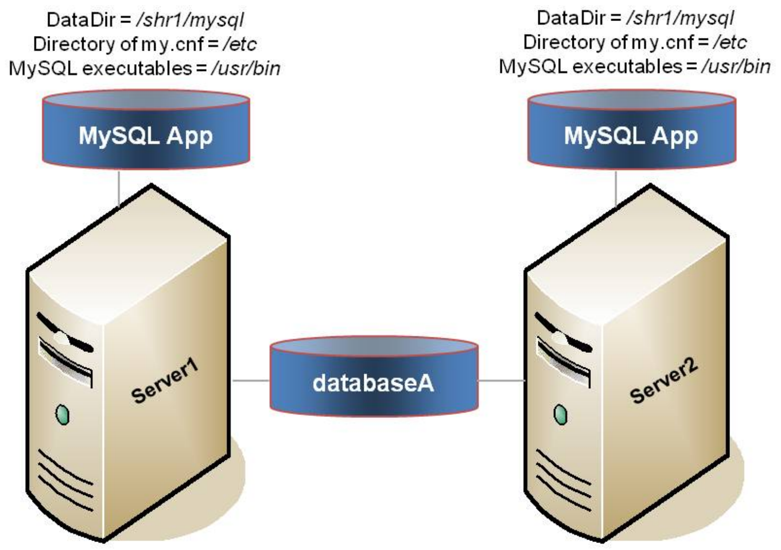
Example 1 - Active/Standby Configuration

Example 2 - Active/Active Configuration

Active - Standby Configuration

This section provides an example of an active/standby configuration. In this configuration, Server 1 is considered active because it has exclusive access to the database. Server 2 does other processing. If Server 1 fails, Server 2 gains access to the database, and LifeKeeper re-establishes the database operations.

Figure 1. Active/Standby Configuration, Example 1


DataDir = /shr1/mysql
Directory of my.cnf = /etc
MySQL executables = /usr/bin
MySQL App
Server1
databaseA
DataDir = /shr1/mysql
Directory of my.cnf = /etc
MySQL executables = /usr/bin
MySQL App
Server2

Configuration Notes:

- Both servers use the MySQL data directory (which includes the database (databaseA)) on a shared disk.
- The path to the MySQL data directory is the same on both servers.
- The my.cnf configuration file is located on a local disk in */etc.*
- The MySQL executables are located on a local drive on each server in */usr/bin*.
- Server 2 cannot access files and directories on the shared disk while Server 1 is active.

Creating a resource hierarchy on Server 1:

Server:	Server1
Directory of *my.cnf* File Location:	*/etc*
Directory of MySQL Executables Location:	*/usr/bin*
Database Tag	mysql-on-server1

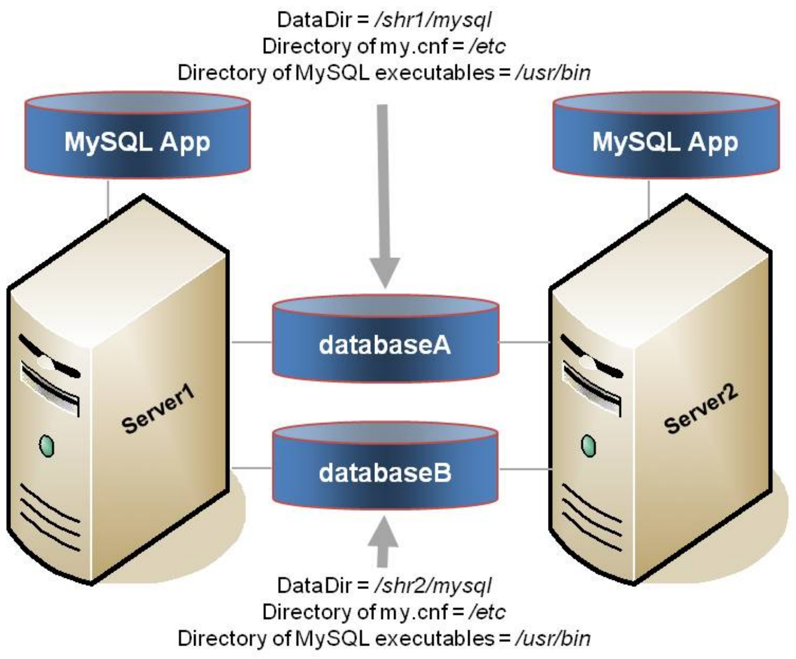
Extending a resource hierarchy to Server 2:

Template Server:	Server1
Tag to Extend	mysql-on-server1
Target Server	Server2
Target Priority:	10
Directory of *my.cnf* File Location:	*/etc*
Directory of MySQL Executables Location:	*/usr/bin*
Database Tag	mysql-on-server2

Active - Active Configuration

An active/active configuration consists of two or more servers actively running a different database instance with each serving as a backup for each other. The database instances must be on different shared physical disks. For LifeKeeper configurations supporting multiple MySQL database instances (of the same or different versions), SIOS recommends that the mysqld Group feature be used for versions of MySQL that support this feature. For these configurations, the *my.cnf* configuration file will reside in */etc*. For MySQL versions that do not support the mysqld Group feature, the *my.cnf* configuration file must reside in the MySQL data directory shared file system for each database instance (e.g. in Figure 2 below, */shr1/mysql* and */shr2/mysql*).

Figure 2. Active/Active Configuration, Example 2


DataDir = /shr1/mysql
Directory of my.cnf = /etc
Directory of MySQL executables = /usr/bin
MySQL App
MySQL App
databaseA
databaseB
Server1
Server2
DataDir = /shr2/mysql
Directory of my.cnf = /etc
Directory of MySQL executables = /usr/bin

Configuration Notes:

- Each server uses a different MySQL data directory (which includes the database instances (database A and database B) on different shared disks
- The path to the MySQL data directory is different for each instance defined on the server.
- The *my.cnf* configuration file is located in */etc* and contains mysqld group sections for each database instance. Each section defines a unique MySQL data directory, port and socket for that database instance.   The *my.cnf* configuration file must be kept in sync on all nodes in the cluster. For systems running versions of MySQL that do not support mysqld Groups, the *my.cnf* configuration file for each of the database instances is located on the shared drive in the data directory for the database instance. Each configuration file defines a unique MySQL data directory, port and socket definition for that database instance.
- The MySQL executables are located on a local drive on each server in */usr/bin*.
- Initially, Server 1 runs databaseA and Server 2 runs databaseB. In a switchover situation, one server can run both databases.

Creating the first resource hierarchy on Server 1:

Server:	Server1
Directory of *my.cnf* File Location:	*/etc*
Directory of my MySQL Executables Location:	*/usr/bin*
Database Tag:	mysql-shared.example.instance1

Extending the first resource hierarchy to Server 2:

Template Server:	Server1
Tag to Extend:	mysql-shared.example.instance1
Target Server:	Server2
Target Priority:	10
Directory of *my.cnf* File Location:	*/etc*
Directory of my MySQL Executables Location:	*/usr/bin*
Database Tag:	mysql-shared.example.instance1

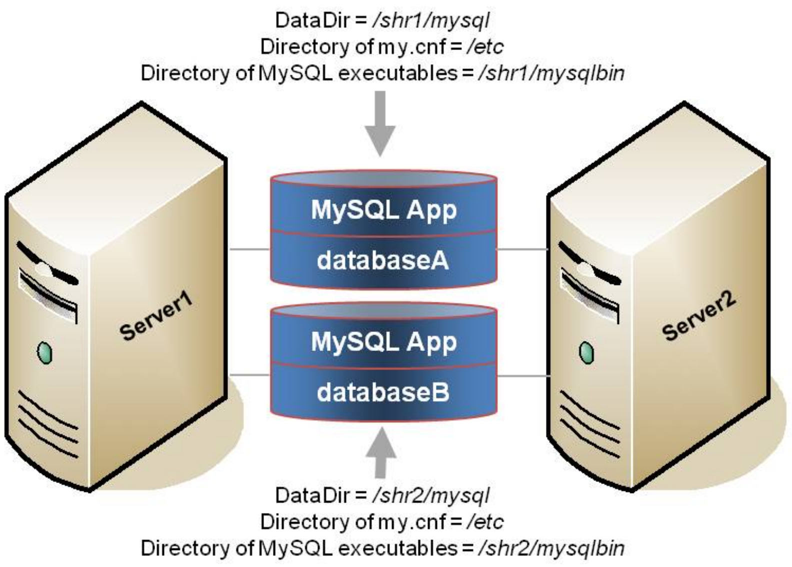
Creating the second resource hierarchy on Server 2:

Server:	Server2
Directory of *my.cnf* File Location:	*/etc*
Directory of MySQL Executables Location:	*/usr/bin*
Database Tag:	mysql-shared.example.instance2

Extending the second resource hierarchy to Server 1:

Template Server:	Server2
Tag to Extend:	mysql-shared.example.instance2
Target Server:	Server1
Target Priority:	10
Directory of *my.cnf* File Location:	*/etc*
Directory of MySQL Executables Location:	*/usr/bin*
Database Tag:	mysql-shared.example.instance2

Figure 3. Active/Active Configuration, Example 2


DataDir = /shr1/mysql
Directory of my.cnf = /etc
Directory of MySQL executables = /shr1/mysqlbin
MySQL App
databaseA
MySQL App
databaseB
Server1
Server2
DataDir = /shr2/mysql
Directory of my.cnf = /etc
Directory of MySQL executables = /shr2/mysqlbin

Configuration Notes:

- Each server uses a different MySQL data directory (which includes the database instances (database A and database B) on different shared disks
- The path to the MySQL data directory is different for each database instance defined on the server.
- The *my.cnf* configuration file is located in */etc* and contains mysqld group sections for each database instance. Each section defines a unique MySQL data directory, port and socket for that database instance. The my.cnf configuration file must be kept in sync on all nodes in the cluster. For systems running versions of MySQL that do not support mysqld Groups, the *my.cnf* configuration file for each of the database instances is located on the shared drive in the data directory for the database. Each

configuration file defines a unique MySQL data directory, port and socket definition for that database instance.

- There is a copy of the MySQL executables on each of the shared disks that contains the data directories.
- Initially, Server 1 runs databaseA and Server 2 runs databaseB. In a switchover situation, one server can run both database instances.

Creating the first resource hierarchy on Server 1:

Server:	Server1
Directory of *my.cnf* File Location:	*/etc*
Directory of MySQL Executables Location:	*/shr1/mysqlbin*
Database Tag:	mysql-shared.example.instance1

Extending the first resource hierarchy to Server 2:

Template Server:	Server1
Tag to Extend:	mysql-shared.example.instance1
Target Server:	Server2
Target Priority:	10
Directory of *my.cnf* File Location:	*/etc*
Directory of MySQL Executables Location:	*/shr1/mysqlbin*
Database Tag:	mysql-shared.example.instance1

Creating the second resource hierarchy on Server 2:

Server:	Server2
Directory of *my.cnf* File Location:	*/etc*
Directory of MySQL Executables Location:	*/shr2/mysqlbin*
Database Tag:	mysql-shared.example.instance2

Extending the second resource hierarchy to Server 1:

Template Server:	Server2
Tag to Extend:	mysql-shared.example.instance2
Target Server:	Server1
Target Priority:	10

Directory of *my.cnf* File Location:	*/etc*
Directory of MySQL Executables Location:	*/shr2/mysqlbin*
Database Tag:	mysql-shared.example.instance2

Multiple Database Server Environment

Following are some configuration considerations if you have multiple MySQL database servers and databases:

- If running active/active or muliple MySQL instances (of the same or different versions), please consider using the mysqld Group feature if possible. SIOS recommends using mysqld Groups (mysqld_multi) for multiple MySQL database server configurations.
- If running active/active or multiple instances of MySQL, do not mount a shared file system as */var/lib/mysql*. This causes unexpected shutdown of MySQL servers by the mysql startup command (`safe_mysqld or mysqld_safe`).
- The *my.cnf* file must be stored in the data directory for each of the active/active or multiple servers if not using the mysqld group feature. For configurations using mysqld Groups, the *my.cnf* file should be stored in */etc* and not in the data directory. For more information on LifeKeeper and the mysqld Group feature, see Using mysqld Groups with LifeKeeper.
- Additional port numbers for MySQL must be specified in the */etc/services* file.
- Each MySQL database server must be configured to run on a different port and access a different socket file. These configuration options are specified in the *my.cnf* file in the data directory.
- Each server must be configured to access data from a different shared location (i.e. each server must use a different data directory).

Using mysqld Groups with LifeKeeper

The MySQL Application Recovery Kit supports my.cnf files using the mysqld group feature managed via mysqld_multi. This MySQL feature allows multiple MySQL instances to be easily configured via a single my.cnf file (typically stored in */etc*.) The kit now detects a my.cnf file using the mysqld group format and prompts the administrator to select the number of the mysqld group to be protected. The choice list provided to the administrator is determined by the group numbers defined in the my.cnf file minus any group numbers already being protected by the kit.

In general, it is easier to set up and control multiple MySQL instances using the mysqld group feature, and SIOS recommends that this approach be used when setting up active/active or multiple instance configurations.

my.cnf File

When using the mysqld group feature, the following are imperative:

a. A single my.cnf file should be used for defining mysqld groups for the database instances.

b. The my.cnf file should NOT be placed on shared storage.

c. An exact copy of the my.cnf file needs to exist on each cluster node (*/etc/my.cnf* is ideal).

d. Any changes made to the my.cnf file must be propagated to every node in the LifeKeeper cluster.

The recovery kit uses mysqld_multi commands when it detects the my.cnf file is using mysqld groups. Based on this, you should be able to use mysqld_multi to test your MySQL instance before placing it under control of LifeKeeper.

The following is a relatively complex my.cnf file using mysqld groups that describes two database instances controlled by mysqld_multi. The `mysqld_multi` command (and the MySQL LifeKeeper recovery kit) gives the administrator a lot of options on how things get set up. In the example below, `[mysqld1]` defines a relatively simple MySQL instance that uses most of the default locations for various MySQL directives. The second example `[mysqld55]` moves things around more. The comments will help describe what each section is doing in terms of LifeKeeper's interaction with MySQL.

```
# The following client section defines which username/password combination
will be used for
# LifeKeeper  connections. The username/password combination needs to be
defined in each MySQL
# Database instance that will be d escribed in this my.cnf file.
[client]
user   =  steeleye
password =  password

# This next section describes the default version of mysqld and mysqldadmin
that mysqld_multi
# will use when processing mysqld_multi commands. The username/password combo
defines the
# MySQL account that mysqld_multi will  use when working with the database
instances. This
# username and password combo needs to be d efined in  each MySQL Database
instance that will be
# controlled by mysqld_multi. See how to set up the  multi_admin account in the
MySQL Reference
# Manual, by issuing "mysqld_multi --example".
[mysqld_multi]
mysqld  = /usr/bin/mysqld_safe
mysqladmin = /usr/bin/mysqladmin
user   = multi_admin
password  = password

# The next section defines the first of two MySQL Database instances in this
my.cnf file. Note
# that each section starts  with a [mysqldNN] where NN is the mysqld group num-
ber (or instance).
# Each group name must have a number. There are a  number of directives that
the LifeKeeper MySQL
# Recovery Kit will be looking for in these sections.
[mysqld1]
```

```
datadir  = /s11/mysql-data5077  # Defines where the data files for the
instance will live. For
   #  LifeKeeper, this directory must be on LifeKeeper protected
   #  (shared or replicated) storage.
mysqld  = /usr/bin/mysqld_safe  # Defines specifically which mysqld command
will be used for
   #  starting the instance. This one is using the
   #  default  mysqld_safe that came with the distribution.
socket=/s11/mysql-data5077/moe.socket # Defines the location of the socket for
this instance.
   #  If the socket is not on LifeKeeper protected storage, it
   #  needs to be defined in exactly the same place on each
   #  node in the cluster and be owned by the "user" defined
   #  below.
port  =  3307  # Each instance needs its own, unique TCP/IP port.
pid-file = /var/run/mysqld/mysqld.pid # The pid-file can be on LifeKeeper pro-
tected or
   #  non-LifeKeeper protected storage.
log-error= /var/log/mysqld.log  # Location of the MySQL error log for this
instance. Can be
   #  on LifeKeeper protected or non-LifeKeeper protected
   #  storage.
user  = mysql  # The Linux user name that will run the MySQL processes.

# The next section defines the more complicated of the two MySQL instances.
Instance "55" is not
# using the default MySQL that  came with the Linux distribution as it is
using the 5.5.12 version
# of MySQL that was installed from source. The binaries for this  version were
installed onto shared
# storage, and the binary directory is LifeKeeper protected.
[mysqld55]
datadir = /s11/mysql-data5512  # Same as above; this instance uses a different
data
   #  d irectory, and  this directory is on LifeKeeper
   #  protected storage.
mysqld  =/s11/mysql5512/bin/mysqld_safe # For this instance, a different ver-
sion of mysqld_safe
   #  is used; the  one that is included with 5.5.12.
socket=/s11/mysql-misc5512/larry.socket # This instance has the socket on
LifeKeeper protected
   #  storage, but not  in the default location (datadir).
port  = 3308  # This instance has a unique TCP/IP port as well.
pid-file = /var/run/mysqld/mysqld55.pid # This instance's pid-file is not on
LifeKeeper protected
   #  storage.
log-error = /var/log/mysqld55.log  # This instance's log-error (error log) is
not on
   #  LifeKeeper protected  storage.
log-bin  = /s11/mysql-log5512/larry  # The log-bin directive specifies where
the binary
```

```
   #  transaction logs are  located for this instance.
   #  These logs must be on LifeKeeper  protected storage
   #  (the recovery kit will enforce this). By default,
   #  these logs are in the datadir.
user  = mysql  # The Linux user name that will run the MySQL processes.
```

When describing both sets up of [mysqld<N>] for multi instance and [mysqld] for single instance, the set up for single instance must be described at the last part.

Example:

```
 [mysqld1]
 (set up for mysqld1)

 [mysqld2]
 (set up for mysqld2)

 [mysqld55]
 (set up for mysqld55)

 [mysqld]
 (set up for mysqld for single instance )
```

mysqld_multi Commands

For this example, issuing the mysql command

```
#  mysqld_multi start 1
```

would start the mysqld group 1 instance defined in my.cnf as [mysqld1], assuming all of the LifeKeeper protected resources that it depends on were in service on one of the LifeKeeper nodes.

Issuing the mysql command

```
# mysqld_multi report 1
```

would report on the status of this instance (e.g. running or not running). Once this instance is running, creating a resource for it in LifeKeeper should be easy.

To get more information on setting up a mysqld_multi style my.cnf file, issue the command

```
# mysqld_multi --example
```

Using Network Attached Storage

There are a couple of special considerations to take into account when configuring LifeKeeper to use an NFS file server (Network Attached Storage) as cluster storage.

Use the NAS Recovery Kit

The optional Network Attached Storage (NAS) recovery kit is required when using an NFS server as a shared storage array with LifeKeeper for Linux. Install the NAS recovery kit (and a license) on each cluster node. See the NAS Recovery Kit documentation for more details.

Possible Error Message

When using Network Attached Storage (NAS) with MySQL, you may experience MySQL instances not restarting following a failover due to a system crash. The MySQL error log should indicate the cause of the error.

MySQL 5.0

```
110523 22:10:58  mysqld started
InnoDB: Unable to lock ./ibdata1, error: 11
InnoDB: Check that you do not already have another mysqld process
InnoDB: using the same InnoDB data or log files.
110523 22:10:58  InnoDB: Retrying to lock the first data file
InnoDB: Unable to lock ./ibdata1, error: 11
InnoDB: Check that you do not already have another mysqld process
InnoDB: using the same InnoDB data or log files.
```

MySQL 5.5

```
110524 10:52:20 InnoDB: The InnoDB memory heap is disabled
110524 10:52:20 InnoDB: Mutexes and rw_locks use GCC atomic builtins
110524 10:52:20 InnoDB: Compressed tables use zlib 1.2.3
110524 10:52:20 InnoDB: Initializing buffer pool, size = 128.0M
110524 10:52:20 InnoDB: Completed initialization of buffer pool
InnoDB: Unable to lock ./ibdata1, error: 11
InnoDB: Check that you do not already have another mysqld process
InnoDB: using the same InnoDB data or log files.
110524 10:52:20  InnoDB: Retrying to lock the first data file
InnoDB: Unable to lock ./ibdata1, error: 11
InnoDB: Check that you do not already have another mysqld process
InnoDB: using the same InnoDB data or log files.
```

This indicates that the MySQL `mysqld` process has set an NFS lock on the file "*ibdata1*" on the NFS file system that is being controlled by LifeKeeper. The lock was not cleared by the system crash, so LifeKeeper is unable to bring the MySQL instance back into service. MySQL thinks that some other process is using the *ibdata1* file.

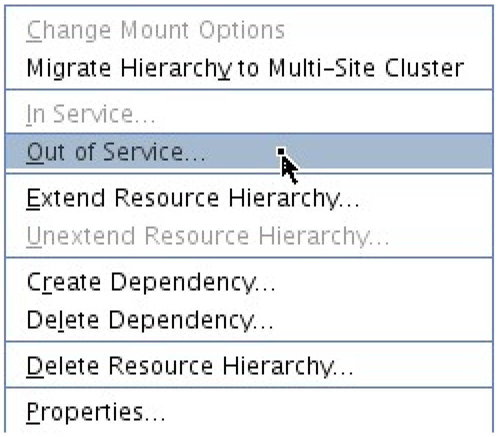
Solution

To fix this, mount the NFS file system that will hold *ibdata1* with the "nolock" NFS option before the File System resource is created. By default, NFS allows file locks to be set. If the "nolock" option is used before resource creation, LifeKeeper will pick up this option and use it each time it brings the file system resource in service. Since LifeKeeper will be controlling access (from the cluster nodes) to the file system containing *ibdata1*, the lock is not typically critical. The NFS mount options used during testing were "`rw,sync,tcp,nfsvers=3,nolock`".

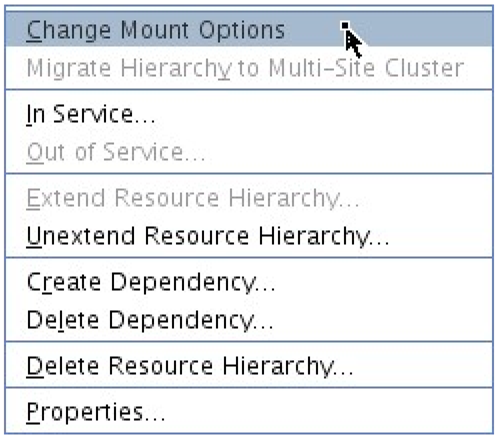
It is not necessary to use the "nolock" on other file systems used by the MySQL resource hierarchy such as the file system where the MySQL binaries are located.

If the NAS File System resource has already been created without the "nolock" option set, use the following procedure to change the mount option:

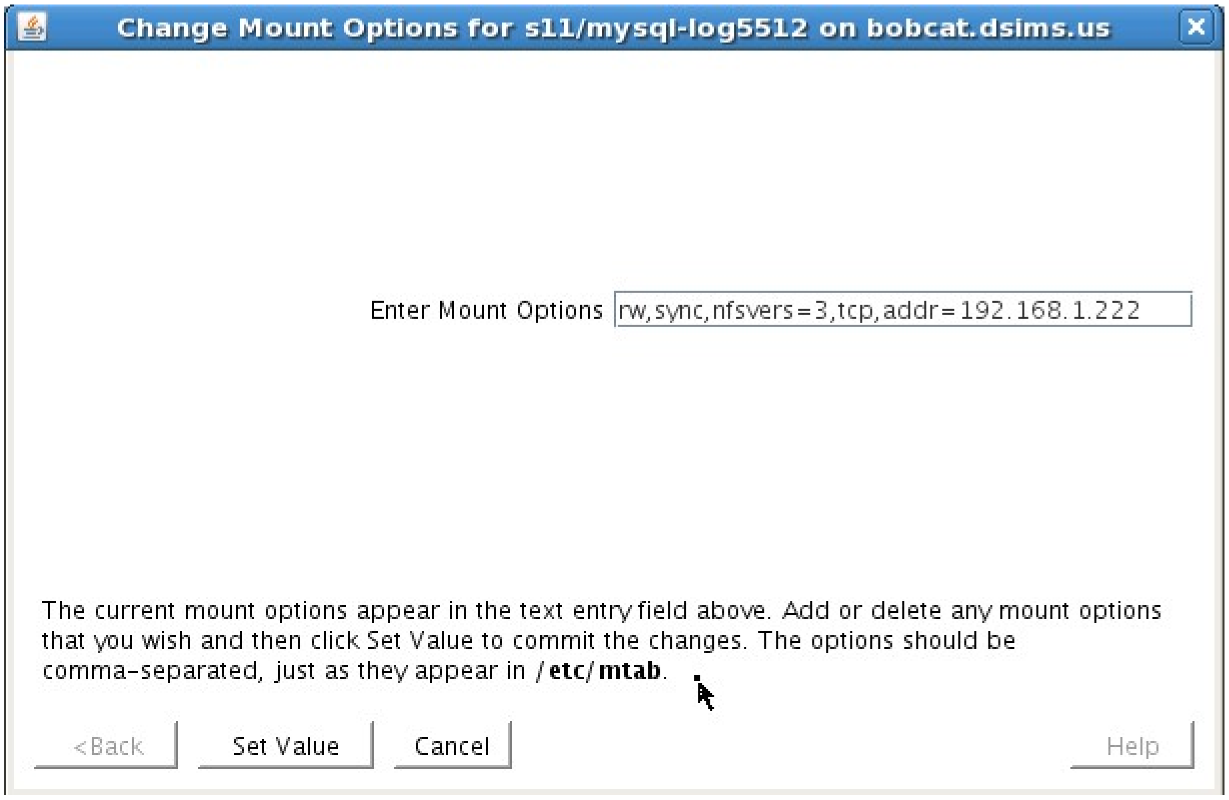
1. Using the LifeKeeper GUI, take the file system resource that needs to be changed out of service. This can be done from the LifeKeeper GUI putting the pointer on the file system resource and doing a right mouse click, and select **Out of Service** from the drop-down menu. This action may take parent resources out of service as well.


Change Mount Options
Migrate Hierarchy to Multi-Site Cluster
In Service...
Out of Service...
Extend Resource Hierarchy...
Unextend Resource Hierarchy...
Create Dependency...
Delete Dependency...
Delete Resource Hierarchy...
Properties...

2. Confirm the **Out of Service** action and allow the process to complete.
3. Once the file system resource is out of service, you can put the pointer on the resource and do another right mouse click, and from the drop-down menu select **Change Mount Options**.


Change Mount Options
Migrate Hierarchy to Multi-Site Cluster
In Service...
Out of Service...
Extend Resource Hierarchy...
Unextend Resource Hierarchy...
Create Dependency...
Delete Dependency...
Delete Resource Hierarchy...
Properties...

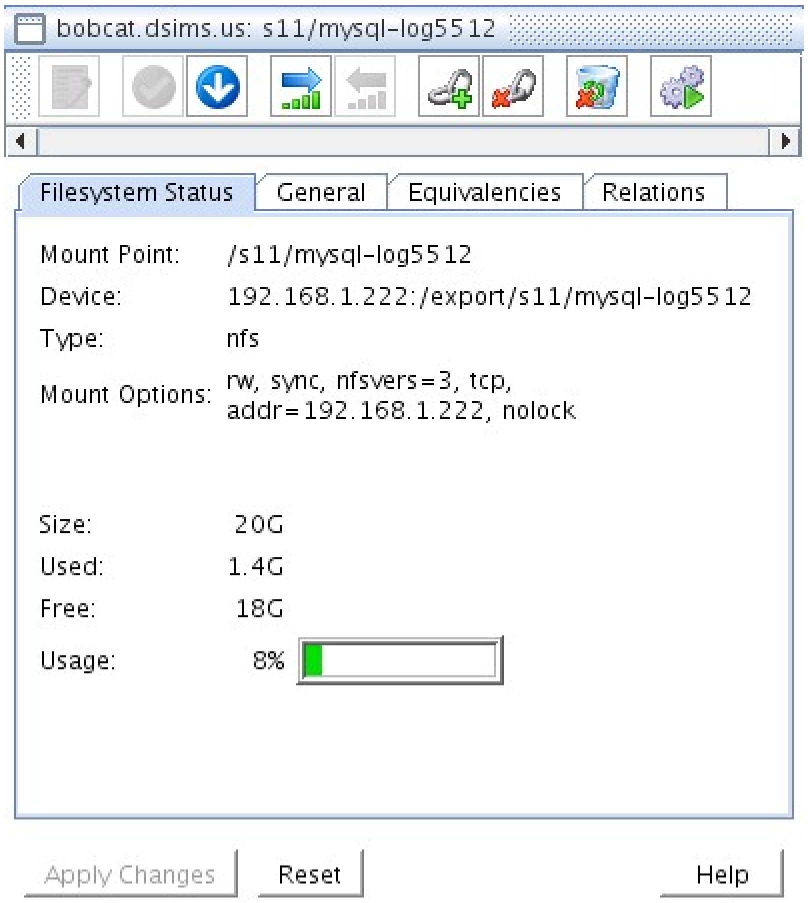
4. In the popup window, add **nolock** to the line of options, and click **Set Value**. You will need to repeat steps 3 and 4 for each node in the cluster.


Change Mount Options for s11/mysql-log5512 on bobcat.dsims.us

Enter Mount Options: rw,sync,nfsvers=3,tcp,addr=192.168.1.222

The current mount options appear in the text entry field above. Add or delete any mount options that you wish and then click Set Value to commit the changes. The options should be comma-separated, just as they appear in **/etc/mtab**.

5. Bring the NAS File System resource back in service by doing a right mouse click, and selecting **In Service**.
6. The File System resource's property panel should now reflect that "nolock" is one of the current mount options.



Mount Point: /s11/mysql-log5512
Device: 192.168.1.222:/export/s11/mysql-log5512
Type: nfs
Mount Options: rw, sync, nfsvers=3, tcp, addr=192.168.1.222, nolock

Size: 20G
Used: 1.4G
Free: 18G
Usage: 8%

Considerations on MySQL use in Systemd environments

If MySQL (version 5.7.6 or later) is installed on a OS distribution adopting systemd, the mysqld_safe and mysqld_multi commands are not installed and thus unavailable for LifeKeeper use. In these environments, LifeKeeper will use the systemctl command to start and stop the MySQL service.

Set up MySQL referring to the article "Managing MySQL Server with systemd". Specially set up the PID File as this is required for Systemd and LifeKeeper. Systemd MySQL set up and my.cnf set up must be the same on all nodes.

The following set up items must be defined for resource creation.

Set up item	Value to set
Location of my.cnf	The full path of the directory that contains the my.cnf file used when starting the MySQL service with the systemctl command
Location of MySQL executables	The full path of the directory where the mysqladmin command is installed

Notes: The setting of "Location of my.cnf" is used inside LifeKeeper to read the settings in my.cnf. The value

set up here is not used by the systemctl startup command. In the case where the my.cnf file is in a different path from the default, set up MySQL for Systemd correctly and make sure the MySQL service starts and stops as expected.
Also, the "include" directive is not supported. All the configuration information must be described in a single my.cnf file.

Chapter 3: Installation

Installing/Configuring MySQL with LifeKeeper

LifeKeeper Configuration Tasks

You can perform the following configuration tasks from the LifeKeeper GUI. The following four tasks are described in this section, as they are unique to a MySQL resource instance, and different for each Recovery Kit.

- Create a Resource Hierarchy. Creates an application resource hierarchy in your LifeKeeper cluster.
- Delete a Resource Hierarchy. Deletes a resource hierarchy from all servers in your LifeKeeper cluster.
- Extend a Resource Hierarchy. Extends a resource hierarchy from the primary server to a backup server.
- Unextend a Resource Hierarchy. Unextends (removes) a resource hierarchy from a single server in the LifeKeeper cluster.

The following tasks are described in the Administration section within the SPS for Linux Technical Documentation because they are common tasks with steps that are identical across all Recovery Kits.

- Create a Resource Dependency. Creates a parent/child dependency between an existing resource hierarchy and another resource instance and propagates the dependency changes to all applicable servers in the cluster.
- Delete a Resource Dependency. Deletes a resource dependency and propagates the dependency changes to all applicable servers in the cluster.
- In Service. Brings a resource hierarchy into service on a specific server.
- Out of Service. Takes a resource hierarchy out of service on a specific server.
- View/Edit Properties. View or edit the properties of a resource hierarchy on a specific server.

Note: Throughout the rest of this section, we explain how to configure your Recovery Kit by selecting certain tasks from the **Edit** menu of the LifeKeeper GUI. You can also select each configuration task from the toolbar. You can also right-click a global resource in the **Resource Hierarchy Tree** (left-hand pane) of the status display window to display the same drop-down menu choices as the **Edit** menu.

You can also right-click a resource instance in the **Resource Hierarchy Table** (right-hand pane) of the status display window to perform all the configuration tasks, except **Creating a Resource Hierarchy**, depending on the state of the server and the particular resource.

Creating a MySQL Resource Hierarchy

IMPORTANT:

In a LifeKeeper cluster environment where the MySQL data directory (datadir) files are on a shared disk, you must make sure that the shared file system is mounted on the primary/template server. If the file system resource is created first, the shared file system MUST be mounted on the same mount point on each server. It is also important to remember that a working communication path (i.e. heartbeat) is required before you can create your resource. The MySQL data directory can exist on shared, replicated or network attached storage.

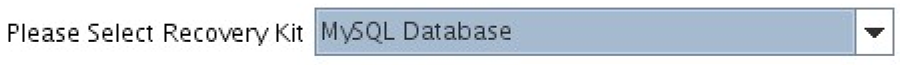
To create a resource instance from the primary server, you should complete the following steps:

1. From the LifeKeeper GUI menu, select **Edit**, then **Server**. From the drop-down menu, select **Create Resource Hierarchy**.

   If you wish to change a selection you have already entered or encounter an error message during any step in the creation of your MySQL resource hierarchy, you will generally be able to back up and change your selection or make corrections (assuming the **Back** button is enabled).

   Important: The MySQL database server daemon (mysqld) for the MySQL instance you want to protect must be running when you create the resource.

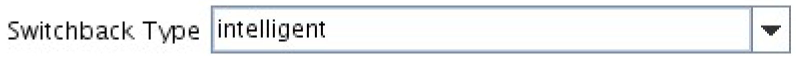
   A dialog box will appear with a drop-down menu listing all recognized Recovery Kits installed within the cluster. Select **MySQL Database** from the drop-down menu.

   
   Please Select Recovery Kit MySQL Database

   Click **Next**.

   If you click the **Cancel** button at any time during the sequence of creating your hierarchy, LifeKeeper will cancel the entire creation process.

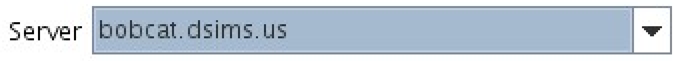
2. Select the **Switchback Type.** This dictates how the MySQL instance will be switched back to this server when it comes back into service after a failover to the backup server. You can choose either ***intelligent*** or ***automatic***. Intelligent switchback requires administrative intervention to switch the instance back to the primary/original server. Automatic switchback means the switchback will occur as soon as the primary server comes back on line and reestablishes LifeKeeper communication paths.

   
   Switchback Type intelligent

The switchback type can be changed later, if desired, from the **General** tab of the **Resource Properties** dialog box.

Click **Next**.

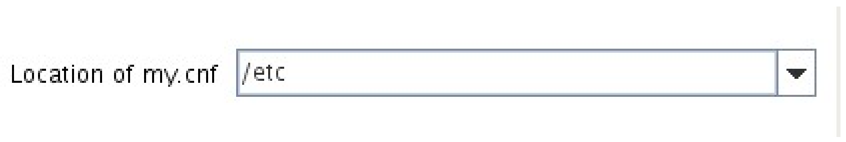
3. Select the **Server** where you want to place the MySQL database instance (typically this is referred to as the primary or template server). All the servers in your cluster are included in the drop-down menu.



Server: bobcat.dsims.us

Click **Next** to proceed to the next dialog box.

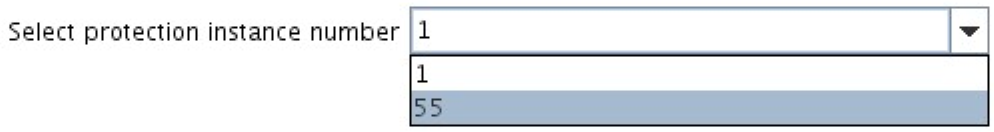
4. Select or enter the **Location of my.cnf**. This is the full path name (excluding the file name) where the MySQL configuration file (*my.cnf*) is located.



Location of my.cnf: /etc

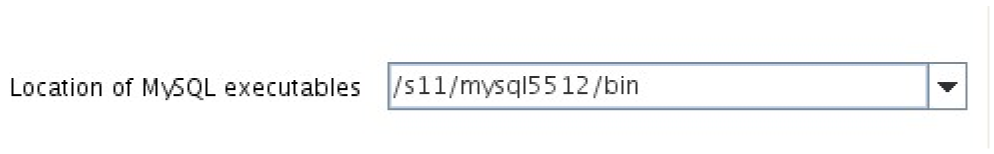
Click **Next** to proceed to the next dialog box.

5. Select the **Protection Instance Number** if you have a mysqld_multi style my.cnf file. If you are using a more traditional style my.cnf file, you will not see this screen.



Select protection instance number: 1

6. Select or enter the **Location of MySQL executables** location**.** This is the full path name of the binaries used to start and monitor the MySQL database server daemon.

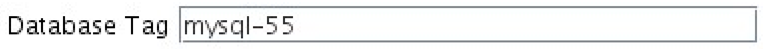


Location of MySQL executables: /s11/mysql5512/bin

Note: At this point, LifeKeeper will validate that you have provided valid data to create your MySQL resource hierarchy. If LifeKeeper detects a problem with either of this validation, an ERROR will appear on the screen. If the directory paths are valid, but there are errors with the MySQL configuration itself, you may pause to correct these errors and continue with the hierarchy creation.

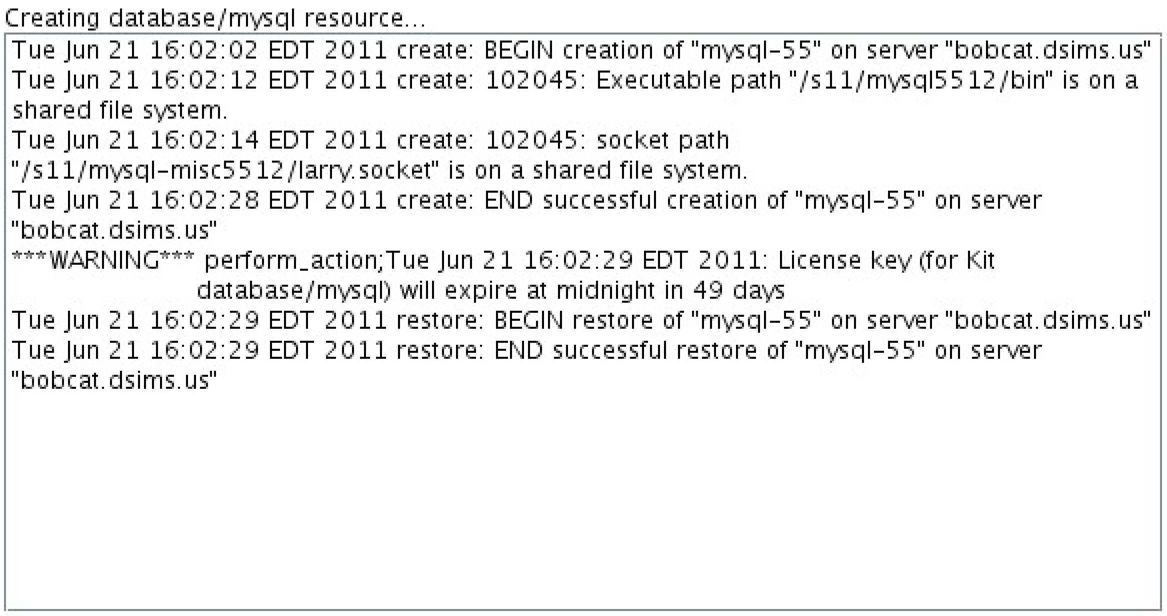
Click **Next** to proceed to the next dialog box.

7. Select or enter the **Database Tag.** This is a tag name given to the MySQL hierarchy. You can select the default or enter your own tag name.



Database Tag mysql-55

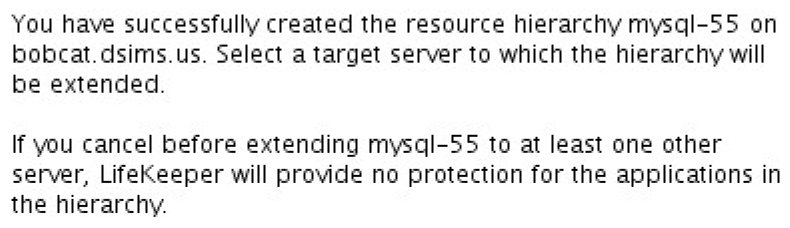
When you click **Create**, the **Create Resource Wizard** will create your MySQL resource.



```
Creating database/mysql resource...
Tue Jun 21 16:02:02 EDT 2011 create: BEGIN creation of "mysql-55" on server "bobcat.dsims.us"
Tue Jun 21 16:02:12 EDT 2011 create: 102045: Executable path "/s11/mysql5512/bin" is on a
shared file system.
Tue Jun 21 16:02:14 EDT 2011 create: 102045: socket path
"/s11/mysql-misc5512/larry.socket" is on a shared file system.
Tue Jun 21 16:02:28 EDT 2011 create: END successful creation of "mysql-55" on server
"bobcat.dsims.us"
***WARNING*** perform_action;Tue Jun 21 16:02:29 EDT 2011: License key (for Kit
                   database/mysql) will expire at midnight in 49 days
Tue Jun 21 16:02:29 EDT 2011 restore: BEGIN restore of "mysql-55" on server "bobcat.dsims.us"
Tue Jun 21 16:02:29 EDT 2011 restore: END successful restore of "mysql-55" on server
"bobcat.dsims.us"
```

Note: The MySQL resource hierarchy should be created successfully at this point.

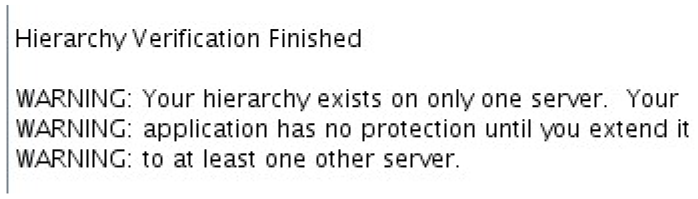
8. Another information box will appear explaining that you have successfully created an MySQL resource hierarchy, and you must **Extend** that hierarchy to another server in your cluster in order to place it under LifeKeeper protection.


You have successfully created the resource hierarchy mysql-55 on bobcat.dsims.us. Select a target server to which the hierarchy will be extended.

If you cancel before extending mysql-55 to at least one other server, LifeKeeper will provide no protection for the applications in the hierarchy.

When you click **Continue**, LifeKeeper will launch the **Pre-Extend Wizard** that is explained in the next section.

If you click **Cancel** now, a dialog box will appear warning you that you will need to come back and extend your MySQL resource hierarchy to another server at some other time to put it under LifeKeeper protection.


```
Hierarchy Verification Finished

WARNING: Your hierarchy exists on only one server.  Your
WARNING: application has no protection until you extend it
WARNING: to at least one other server.
```

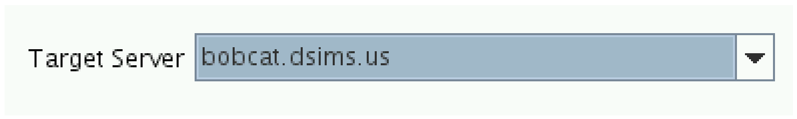
9. Click **Done** to exit.

Deleting a Resource Hierarchy

To delete a resource hierarchy from <u>all</u> the servers in your LifeKeeper environment, complete the following steps:

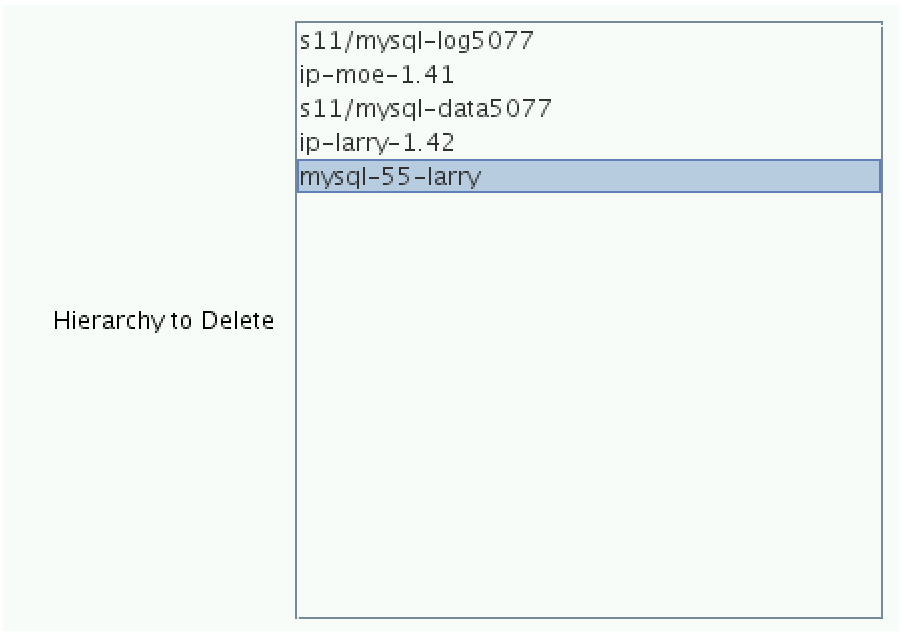
1. From the LifeKeeper GUI menu, select **Edit**, and then **Resource**. From the drop-down menu, select **Delete Resource Hierarchy**.
2. Select the name of the **Target Server** where you will be deleting your MySQL resource hierarchy.

   Note: If you selected the **Delete Resource** task by right-clicking from the right pane on an individual resource instance, or from the left pane on a global resource where the resource is on only one server, this dialog box will not appear.

   
   Target Server bobcat.dsims.us

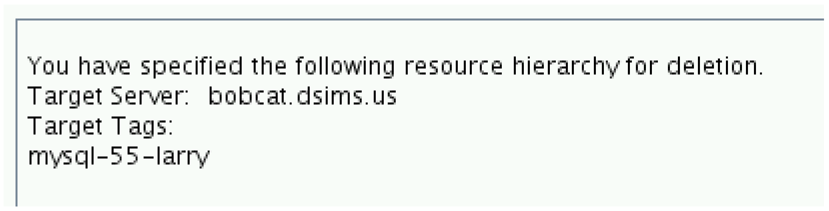
   Click **Next**.
3. Select the **Hierarchy to Delete**. Identify the resource hierarchy you wish to delete, and highlight it.

Note: If you selected the **Delete Resource** task by right-clicking from either the left pane on a global resource or the right pane on an individual resource instance, this dialog will not appear.



Click **Next**.

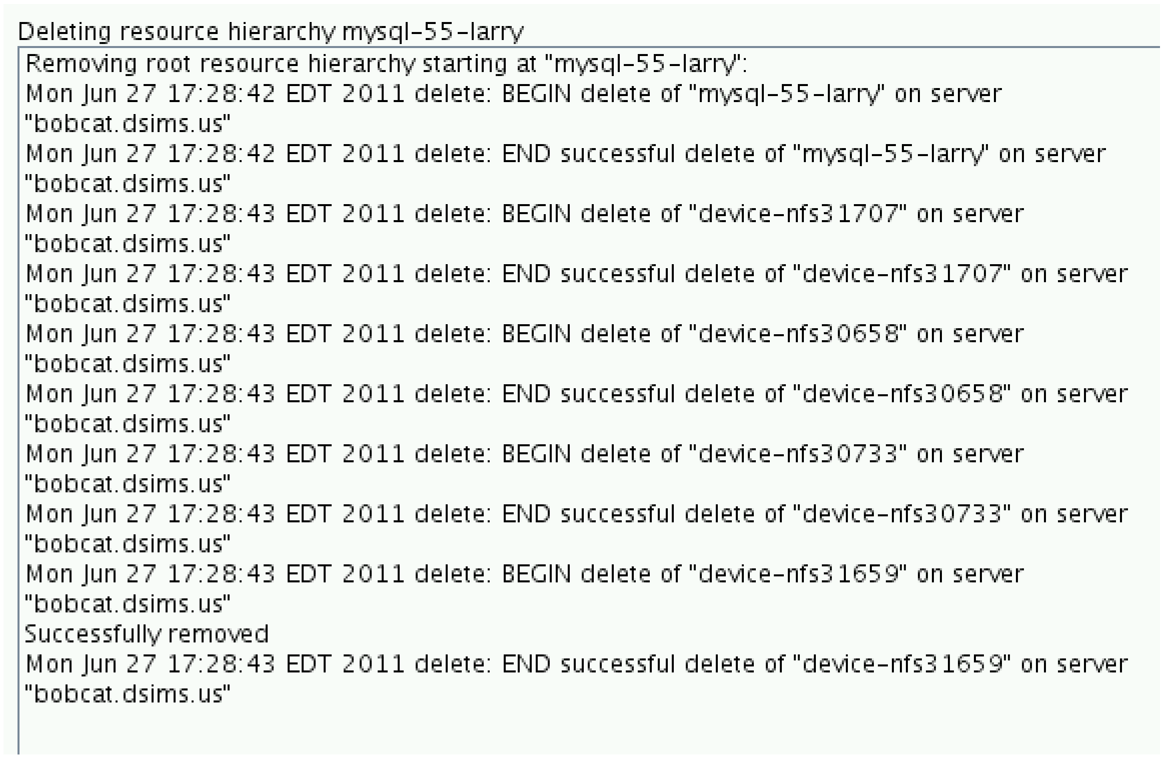
4. An information box appears confirming your selection of the target server and the hierarchy you have selected to delete.



You have specified the following resource hierarchy for deletion.
Target Server: bobcat.dsims.us
Target Tags:
mysql-55-larry

Click **Delete**.

5. Another information box appears confirming that the MySQL resource was deleted successfully.



```
Deleting resource hierarchy mysql-55-larry
Removing root resource hierarchy starting at "mysql-55-larry":
Mon Jun 27 17:28:42 EDT 2011 delete: BEGIN delete of "mysql-55-larry" on server
"bobcat.dsims.us"
Mon Jun 27 17:28:42 EDT 2011 delete: END successful delete of "mysql-55-larry" on server
"bobcat.dsims.us"
Mon Jun 27 17:28:43 EDT 2011 delete: BEGIN delete of "device-nfs31707" on server
"bobcat.dsims.us"
Mon Jun 27 17:28:43 EDT 2011 delete: END successful delete of "device-nfs31707" on server
"bobcat.dsims.us"
Mon Jun 27 17:28:43 EDT 2011 delete: BEGIN delete of "device-nfs30658" on server
"bobcat.dsims.us"
Mon Jun 27 17:28:43 EDT 2011 delete: END successful delete of "device-nfs30658" on server
"bobcat.dsims.us"
Mon Jun 27 17:28:43 EDT 2011 delete: BEGIN delete of "device-nfs30733" on server
"bobcat.dsims.us"
Mon Jun 27 17:28:43 EDT 2011 delete: END successful delete of "device-nfs30733" on server
"bobcat.dsims.us"
Mon Jun 27 17:28:43 EDT 2011 delete: BEGIN delete of "device-nfs31659" on server
"bobcat.dsims.us"
Successfully removed
Mon Jun 27 17:28:43 EDT 2011 delete: END successful delete of "device-nfs31659" on server
"bobcat.dsims.us"
```

6. Click **Done** to exit.

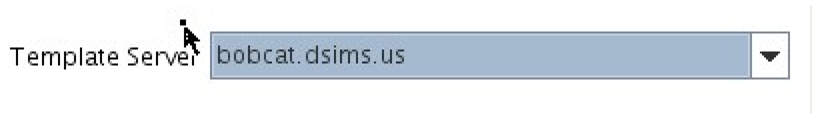
Extending Your Hierarchy

After you have created a hierarchy, you will want to extend that hierarchy to another server in the cluster. There are three possible scenarios to extend your resource instance from the template server to a target server. The first scenario is when you "**Continue**" from creating the resource into extending that resource to another server. The second scenario is when you enter the **Extend Resource Hierarchy** task from the edit menu as shown below. The third scenario is when you right-click on an unextended hierarchy in either the left or right pane. Each scenario takes you through the same dialog boxes (with a few exceptions, which are clearly detailed below).

1. If you are entering the **Extend** wizard from the LifeKeeper GUI menu, select **Edit**, then **Resource**. From the drop-down menu, select **Extend Resource Hierarchy**. This will launch the **Extend Resource Hierarchy** wizard.

2. The first dialog box to appear will ask you select the **Template Server** where your MySQL resource hierarchy is currently in service. It is important to remember that the **Template Server** you select now and the **Tag to Extend** that you select in the next dialog box represent an *in service* resource hierarchy. An error message will appear if you select a resource tag that is not in service on the template server you selected. The drop-down box in this dialog provides the names of all the servers in your cluster.

Note: If you are entering the **Extend Resource Hierarchy** task immediately following the creation of a MySQL resource hierarchy, this dialog box will not appear, since the wizard has already identified the template server in the create stage. This is also the case when you right-click either the MySQL resource icon in the left pane or right-click on the MySQL resource box in the right pane the of the GUI window and choose **Extend Resource Hierarchy**.


Template Server bobcat.dsims.us

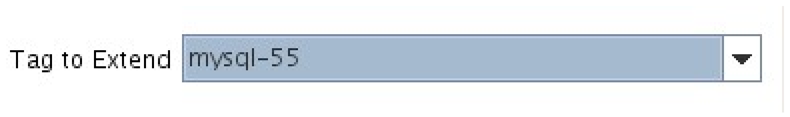
It should be noted that if you click **Cancel** at any time during the sequence of extending your hierarchy, LifeKeeper will cancel the extension process to that particular server. However, if you have already extended the resource to another server, that instance will continue to be in effect until you specifically unextend it.

For example, let us say you have created your resource on Server 1 and extended that resource to Server 2. In the middle of extending the same resource to Server 3, you change your mind and click **Cancel** inside one of the dialog boxes. This will cancel only your action to extend the resource to Server 3, not the extension you created to Server 2. If you want to remove Server 2 from this hierarchy, you must unextend the resource from Server 2.

Click **Next** to proceed to the next dialog box.

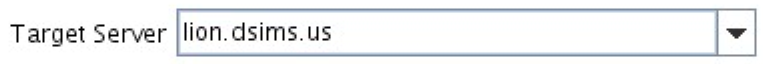
3. Select the **Tag to Extend**. This is the name of the MySQL instance you wish to extend from the template server to the target server. The wizard will list in the drop-down menu all the resources that you have created on the template server, which you selected in the previous dialog box.

   Note: Once again, if you are entering the Extend Resource Hierarchy task immediately following the creation of a MySQL resource hierarchy, this dialog box will not appear, since the wizard has already identified the tag name of your MySQL resource in the create stage. This is also the case when you right-click either the MySQL resource icon in the left hand pane or on the MySQL resource box in the right hand pane of the GUI window and choose **Extend Resource Hierarchy**.

   
   Tag to Extend mysql-55

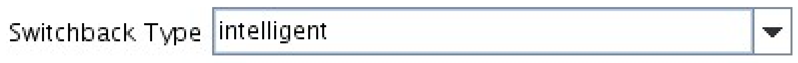
   Click **Next**.

4. Select the **Target Server** where you are extending your MySQL resource hierarchy. The drop-down box provides the names of the servers in your cluster that are not already in the selected hierarchy.

   
   Target Server lion.dsims.us

   Click **Next**.

5. Select the **Switchback Type.** This dictates how the MySQL instance will be switched back to this server when it comes back into service after a failover to the backup server. You can choose either ***intelligent*** or ***automatic***. Intelligent switchback requires administrative intervention to switch the instance back to the primary/original server. Automatic switchback means the switchback will occur as soon as the primary server comes back online and reestablishes LifeKeeper communication paths.

   
   Switchback Type intelligent

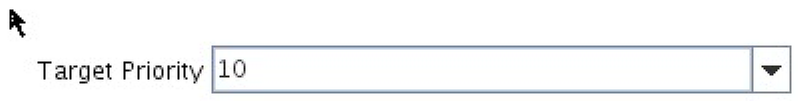
   The switchback type can be changed later, if desired, from the **General** tab of the **Resource Properties** dialog box.

   Click **Next**.

6. Select or enter a **Template Priority**. This is the priority for the MySQL hierarchy on the server where it is currently in service. Any unused priority value from 1 to 999 is valid, where a lower number means a higher priority (1=highest). The extend process will reject any priority for this hierarchy that is already in use by another system. The default value is recommended. **Note:** This selection will appear only for the initial extend of the hierarchy.

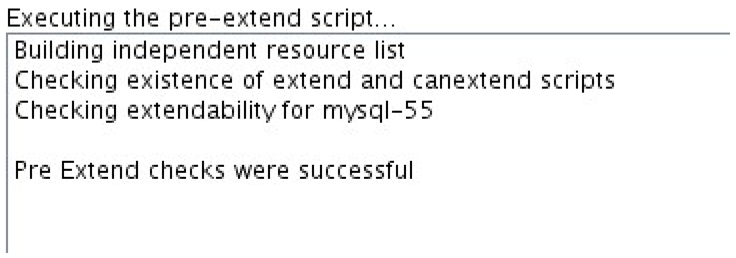
   Click **Next.**

7. Select or enter the **Target Priority**. This is the priority for the new extended MySQL hierarchy relative to equivalent hierarchies on other servers. Any unused priority value from 1 to 999 is valid, indicating a server's priority in the cascading failover sequence for the resource. A lower number means a higher priority (1=highest). Note that LifeKeeper assigns the number "1" to the server on which the hierarchy is created by default. The priorities need not be consecutive, but no two servers can have the same priority for a given resource.

   
   Target Priority 10

   Click **Next**.

8. An information box will appear explaining that LifeKeeper has successfully checked your environment and that all the requirements for extending this MySQL resource have been met. If there were some requirements that had not been met, LifeKeeper would not allow you to select the **Next** button, and the **Back** button would be enabled.



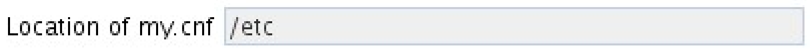
```
Executing the pre-extend script...
Building independent resource list
Checking existence of extend and canextend scripts
Checking extendability for mysql-55

Pre Extend checks were successful
```

If you click **Back**, you can make changes to your resource extension according to any error messages that may appear in the information box.

If you click **Cancel** now, you will need to come back and extend your MySQL resource hierarchy to another server at some other time to put it under LifeKeeper protection.

When you click **Next**, LifeKeeper will launch you into the **Extend Resource Hierarchy** configuration task.

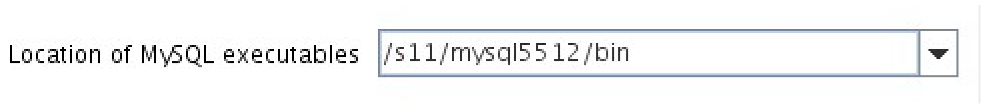
9. This dialog box is for information purposes only. You cannot change the **Location of my.cnf** that appears in the box. The MySQL instance acquired the location information from its configuration file.



Location of my.cnf: /etc

Click **Next**.

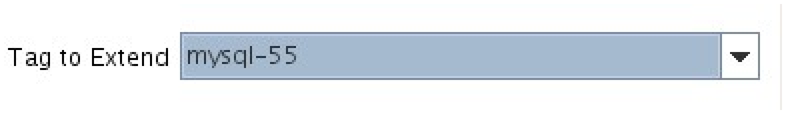
10. Select or enter the **Location of MySQL executables**.This is the full path name of the binaries used to start and monitor the MySQL database server daemon.



Location of MySQL executables: /s11/mysql5512/bin

Click **Next**.

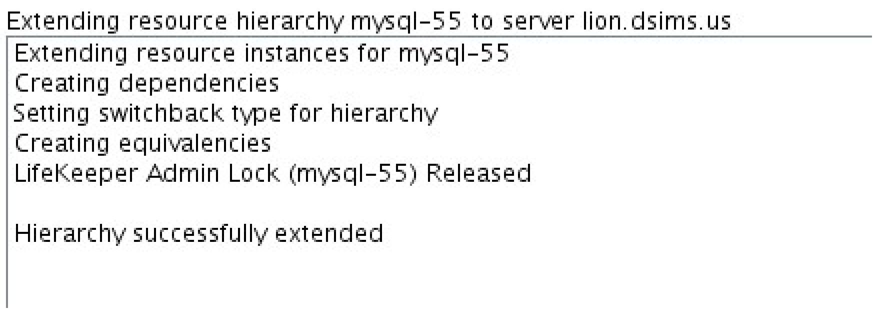
11. Select or enter the **Database Tag.** This is a tag name given to the MySQL hierarchy. You can select the default or enter your own tag name.



Tag to Extend: mysql-55

Click **Extend**.

12. An information box will appear verifying that the extension is being performed.

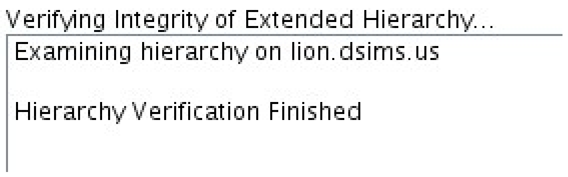


```
Extending resource hierarchy mysql-55 to server lion.dsims.us
Extending resource instances for mysql-55
Creating dependencies
Setting switchback type for hierarchy
Creating equivalencies
LifeKeeper Admin Lock (mysql-55) Released

Hierarchy successfully extended
```

Click **Next Server** if you want to extend the same MySQL resource instance to another server in your cluster. This will repeat the **Extend Resource Hierarchy** operation.

If you click **Finish**, LifeKeeper will verify that the extension of the MySQL resource was completed successfully.

13. If you clicked **Finish**, the following screen appears.



```
Verifying Integrity of Extended Hierarchy...
Examining hierarchy on lion.dsims.us

Hierarchy Verification Finished
```

14. Click **Done** in the last dialog box to exit.

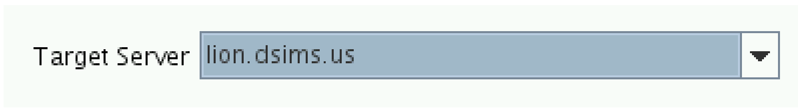
    Note: Be sure to test the functionality of the new instance on *both* servers.

Unextending Your Hierarchy

1. From the LifeKeeper GUI menu, select **Edit**, and **Resource**. From the drop-down menu, select **Unextend Resource Hierarchy**.

2. Select the **Target Server** where you want to unextend the MySQL resource. It cannot be the server where the MySQL resource is currently in service.

   Note: If you selected the **Unextend** task by right-clicking from the right pane on an individual resource instance, this dialog box will not appear.

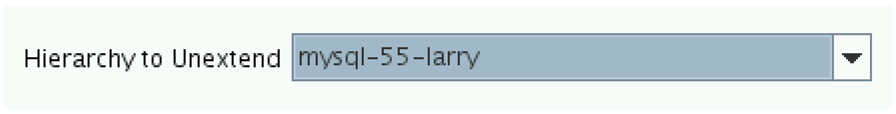


Target Server: lion.dsims.us

   Click **Next**.

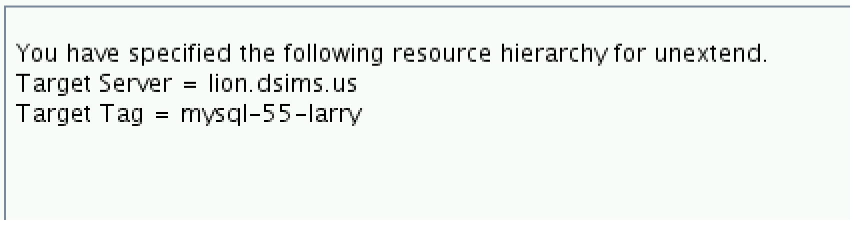
3. Select the MySQL **Hierarchy to Unextend.**

Note: If you selected the **Unextend** task by right-clicking from either the left pane on a global resource or the right pane on an individual resource instance, this dialog will not appear.



Click **Next**.

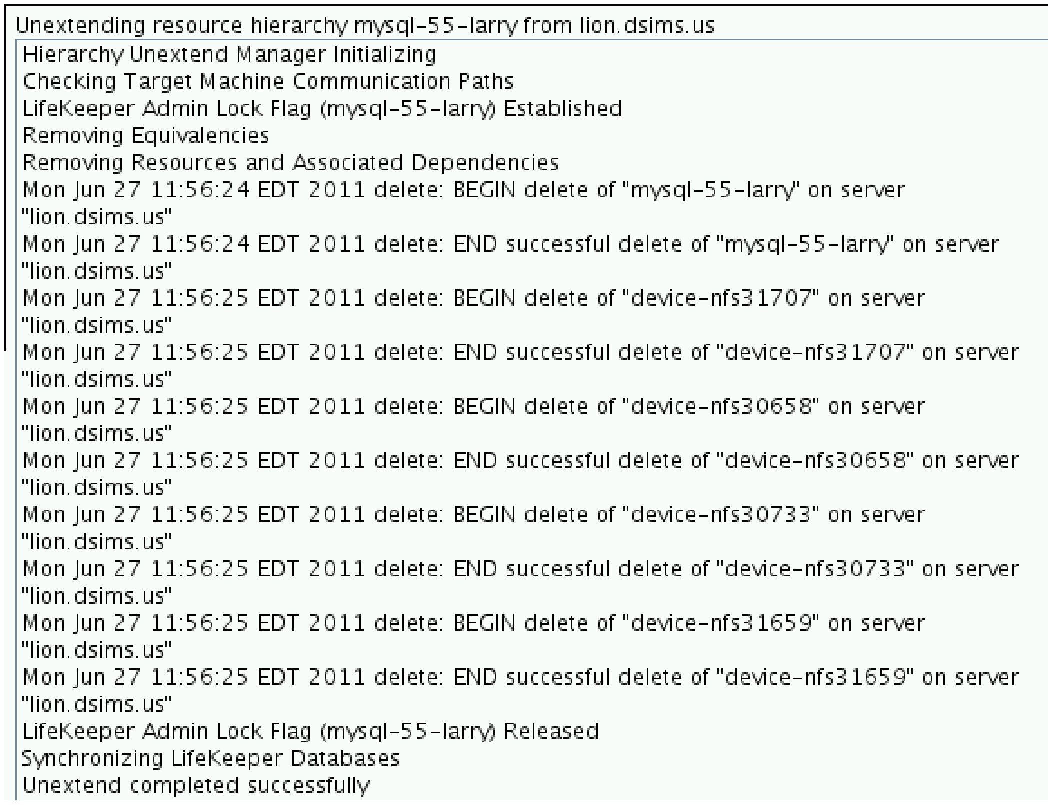
4. An information box appears confirming the target server and the MySQL resource hierarchy you have chosen to unextend.



You have specified the following resource hierarchy for unextend.
Target Server = lion.dsims.us
Target Tag = mysql-55-larry

Click **Unextend**.

5. Another information box appears confirming that the MySQL resource was unextended successfully.



```
Unextending resource hierarchy mysql-55-larry from lion.dsims.us
Hierarchy Unextend Manager Initializing
Checking Target Machine Communication Paths
LifeKeeper Admin Lock Flag (mysql-55-larry) Established
Removing Equivalencies
Removing Resources and Associated Dependencies
Mon Jun 27 11:56:24 EDT 2011 delete: BEGIN delete of "mysql-55-larry" on server
"lion.dsims.us"
Mon Jun 27 11:56:24 EDT 2011 delete: END successful delete of "mysql-55-larry" on server
"lion.dsims.us"
Mon Jun 27 11:56:25 EDT 2011 delete: BEGIN delete of "device-nfs31707" on server
"lion.dsims.us"
Mon Jun 27 11:56:25 EDT 2011 delete: END successful delete of "device-nfs31707" on server
"lion.dsims.us"
Mon Jun 27 11:56:25 EDT 2011 delete: BEGIN delete of "device-nfs30658" on server
"lion.dsims.us"
Mon Jun 27 11:56:25 EDT 2011 delete: END successful delete of "device-nfs30658" on server
"lion.dsims.us"
Mon Jun 27 11:56:25 EDT 2011 delete: BEGIN delete of "device-nfs30733" on server
"lion.dsims.us"
Mon Jun 27 11:56:25 EDT 2011 delete: END successful delete of "device-nfs30733" on server
"lion.dsims.us"
Mon Jun 27 11:56:25 EDT 2011 delete: BEGIN delete of "device-nfs31659" on server
"lion.dsims.us"
Mon Jun 27 11:56:25 EDT 2011 delete: END successful delete of "device-nfs31659" on server
"lion.dsims.us"
LifeKeeper Admin Lock Flag (mysql-55-larry) Released
Synchronizing LifeKeeper Databases
Unextend completed successfully
```

6. Click **Done** to exit.

Chapter 4: Administration

Testing Your Resource Hierarchy

You can test your MySQL resource hierarchy by initiating a manual switchover. This will simulate a failover of a resource instance from the primary server to the backup server.

Performing a Manual Switchover from the GUI

Performing a Manual Switchover from the GUI

You can test your MySQL resource hierarchy by initiating a manual switchover. This will simulate a failover of a resource instance from the primary server to the backup server.

Performing a Manual Switchover from the GUI

You can initiate a manual switchover from the LifeKeeper GUI by selecting **Edit**, **Resource** and **In Service** from the drop-down menu. For example, an **In-Service** request executed on a backup server causes the application hierarchy to be placed in service on the backup server and taken out of service on the primary server. At this point, the original backup server is now the primary server and original primary server has now become the backup server.

If you execute the **Out-of-Service** request, the application is taken out of service without bringing it in service on the other server.

LifeKeeper does not regulate or control internal operations such as rollbacks and backing up archives. Tape archiving and restoration are the responsibility of the application administrator.

Recovery Operations

When the primary server fails, the MySQL Recovery Kit software performs the following tasks:

- Mounts the file system(s) - shared or replicated - on the backup server
- Starts the daemon processes related to MySQL

Chapter 5: Troubleshooting

Common Error Messages

This section provides a list of messages that you may encounter while creating and extending an SPS MySQL resource hierarchy or removing and restoring a resource. Where appropriate, it provides an additional explanation of the cause of an error and necessary action to resolve the error condition.

Messages from other SPS components are also possible. In these cases, please refer to the Message Catalog (located on our Technical Documentation site under “Search for an Error Code”) which provides a listing of all error codes, including operational, administrative and GUI, that may be encountered while using SIOS Protection Suite for Linux and, where appropriate, provides additional explanation of the cause of the error code and necessary action to resolve the issue. This full listing may be searched for any error code received, or you may go directly to one of the individual Message Catalogs for the appropriate SPS component.

MySQL Specific Error Messages

Note: In the Error Message column, a word in quotations and all capital letters refers to the name of a resource on the server (for example, “SERVER” might actually be a server named “Server1”).

Error Number	Error Message
102001	Usage: “SCRIPT NAME” sysname dbvarname cnfpath exepath instance
102002	Usage: “SCRIPT NAME” cnfpath
102003	Usage: “SCRIPT NAME” exepath cnfpath
102004	Unable to obtain a valid value for the “socket” variable in “PATH”/my.cnf **Action:** There must be an entry for the “socket” in the 'mysqld' section of the my.cnf configuration file
102005	Unable to obtain a valid value for the “port” in “PATH”/my.cnf **Action:** There must be an entry for the “port” in the 'mysqld' section of the my.cnf configuration file
102006	Unable to obtain the data directory location "PATH" **Action:** Please make sure that the database is running using the socket and port specified.
102007	Must specify the absolute path to the my.cnf configuration file

Error Number	Error Message
102008	Must specify the absolute path to the MySQL executables
102009	The file my.cnf does not exist in the path specified
102010	The MySQL executables do not exist in the path specified
102011	LifeKeeper was unable to start the MySQL database server
102012	LifeKeeper successfully started the MySQL database server
102013	LifeKeeper was unable to stop the MySQL database server
102014	LifeKeeper successfully stopped the MySQL database server
102015	The port "PORT NUMBER" is in use on the target server "SERVER"
102016	The MySQL database server is not running on server "SERVER"
102017	Unable to open the configuration file "PATH"/my.cnf
102018	Unable to get the Data Directory information for resource "TAG" on server "SERVER"
102019	Unable to get the configuration file location information for resource "TAG" on server "SERVER"
102020	Unable to get the executable location information for resource "TAG" on server "SERVER"
102021	The argument for the configuration file path is empty
102022	The argument for the executable path is empty
102023	The path "PATH" for directive "DIRECTIVE" is not on a shared filesystem
102024	Unable to get the information for resource "TAG" on system "SYSTEM"
102025	The MySQL data directory "DATADIR" is already under LifeKeeper protection
102026	The port variables in the file /etc/my.cnf on "SERVER1" and "SERVER2" do not match
102027	The socket variables in the file /etc/my.cnf on "SERVER1" and "SERVER2" do not match
102028	Unable to obtain a valid value for the "user" variable in "PATH"/my.cnf **Action:** There must be a valid entry for the "user" variable in the 'client' section of the my.cnf configuration file
102029	Unable to obtain a valid value for the "password" variable in "PATH"/my.cnf **Action:** There must be a valid entry for the "password" variable in the 'client' section of the my.cnf configuration file
102030	The user variables in the file /etc/my.cnf on "SERVER1" and "SERVER2" do not match
102031	The password variables in the file /etc/my.cnf on "SERVER1" and "SERVER2" do not match
102032	Unable to obtain the pid file location **Action:** There must be an entry for the "pid-file" variable in the 'mysqld' section of the my.cnf configuration file

Error Number	Error Message
102033	Unable to obtain a valid value for the "user" variable in “PATH”/my.cnf **Action:** The OS user must be specified using the "user" variable in the 'mysqld' section of the my.cnf configuration file
102034	WARNING: A my.cnf file exists at "PATH", which may override the values specified in the file at "PATH"/my.cnf.
102035	The mysql system user "USER" does not exist on target server "SERVER"
102036	The mysql system user "USER" uids are different on target server "SERVER1" and template server "SERVER2"
102037	The mysql system user "USER" gids are different on target server "SERVER1" and template server "SERVER2"
102038	LifeKeeper was unable to stop the MySQL database server using a graceful shutdown. Issuing kill for pid(s): "PROCESS ID LIST".
102039	LifeKeeper will ignore failed connection as possible max connections error, due to existence of process pid "PROCESS ID".
102040	The mysql action for resource tag :TAG" returned: "COMMAND OUTPUT".
102041	LifeKeeper was unable to start the MySQL database server using the defaults-file option. Retrying with individual options.
102042	The LifeKeeper "ACTION" action detected the flag "FLAG", and will exit.
102043	END of "ACTION" action on due to a(n) "SIGNAL" signal.
102044	The file my.cnf does not exist in the stored path "PATH".
102045	"DIRECTIVE" path "PATH" is on a shared filesystem.
102046	Starting mysqld daemon with databases from "PATH".