

Starting and Stopping MySQL

Abstract

This is the Starting and Stopping MySQL extract from the MySQL 5.6 Reference Manual.

For legal information, see the [Legal Notices](#).

For help with using MySQL, please visit either the [MySQL Forums](#) or [MySQL Mailing Lists](#), where you can discuss your issues with other MySQL users.

Document generated on: 2019-05-17 (revision: 62085)

Table of Contents

Preface and Legal Notices	v
1 Installing MySQL on Unix/Linux Using Generic Binaries	1
2 Starting the Server for the First Time on Windows	5
3 MySQL Notifier	7
3.1 MySQL Notifier Usage	7
3.2 Setting Up Remote Monitoring in MySQL Notifier	13
4 The Server Shutdown Process	19
5 MySQL Server and Server-Startup Programs	21
5.1 <code>mysqld</code> — The MySQL Server	21
5.2 <code>mysqld_safe</code> — MySQL Server Startup Script	21
5.3 <code>mysql.server</code> — MySQL Server Startup Script	26
5.4 <code>mysqld_multi</code> — Manage Multiple MySQL Servers	28

Preface and Legal Notices

This is the Starting and Stopping MySQL extract from the MySQL 5.6 Reference Manual.

Licensing information—MySQL 5.6. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 5.6, see the [MySQL 5.6 Commercial Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL 5.6, see the [MySQL 5.6 Community Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL NDB Cluster 7.3. This product may include third-party software, used under license. If you are using a *Commercial* release of NDB Cluster 7.3, see the [MySQL NDB Cluster 7.3 Commercial Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of NDB Cluster 7.3, see the [MySQL NDB Cluster 7.3 Community Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL NDB Cluster 7.4. This product may include third-party software, used under license. If you are using a *Commercial* release of NDB Cluster 7.4, see the [MySQL NDB Cluster 7.4 Commercial Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of NDB Cluster 7.4, see the [MySQL NDB Cluster 7.4 Community Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Community release.

Legal Notices

Copyright © 1997, 2019, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and

other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Chapter 1 Installing MySQL on Unix/Linux Using Generic Binaries

Oracle provides a set of binary distributions of MySQL. These include generic binary distributions in the form of compressed `tar` files (files with a `.tar.gz` extension) for a number of platforms, and binaries in platform-specific package formats for selected platforms.

This section covers the installation of MySQL from a compressed `tar` file binary distribution on Unix/Linux platforms. For other platform-specific binary package formats, see the other platform-specific sections in this manual. For example, for Windows distributions, see [Installing MySQL on Microsoft Windows](#). See [How to Get MySQL](#) on how to obtain MySQL in different distribution formats.

MySQL compressed `tar` file binary distributions have names of the form `mysql-VERSION-OS.tar.gz`, where `VERSION` is a number (for example, `5.6.45`), and `OS` indicates the type of operating system for which the distribution is intended (for example, `pc-linux-i686` or `winx64`).

Warning

If you have previously installed MySQL using your operating system native package management system, such as Yum or APT, you may experience problems installing using a native binary. Make sure your previous MySQL installation has been removed entirely (using your package management system), and that any additional files, such as old versions of your data files, have also been removed. You should also check for configuration files such as `/etc/my.cnf` or the `/etc/mysql` directory and delete them.

For information about replacing third-party packages with official MySQL packages, see the related [APT guide](#) or [Yum guide](#).

Warning

MySQL has a dependency on the `libaio` library. Data directory initialization and subsequent server startup steps will fail if this library is not installed locally. If necessary, install it using the appropriate package manager. For example, on Yum-based systems:

```
shell> yum search libaio # search for info
shell> yum install libaio # install library
```

Or, on APT-based systems:

```
shell> apt-cache search libaio # search for info
shell> apt-get install libaio1 # install library
```

To install a compressed `tar` file binary distribution, unpack it at the installation location you choose (typically `/usr/local/mysql`). This creates the directories shown in the following table.

Table 1.1 MySQL Installation Layout for Generic Unix/Linux Binary Package

Directory	Contents of Directory
<code>bin, scripts</code>	<code>mysqld</code> server, client and utility programs
<code>data</code>	Log files, databases
<code>docs</code>	MySQL manual in Info format
<code>include</code>	Include (header) files

Directory	Contents of Directory
<code>lib</code>	Libraries
<code>mysql-test</code>	Test suite
<code>man</code>	Unix manual pages
<code>share</code>	Error messages, dictionary, and SQL for database installation
<code>sql-bench</code>	Benchmarks
<code>support-files</code>	Miscellaneous support files, including sample configuration files

Note

SLES 11: as of MySQL 5.6.37, the Linux Generic tarball package format is EL6 instead of EL5. As a side effect, the MySQL client `bin/mysql` needs `libtinfo.so.5`.

A workaround is to create a symlink, such as `ln -s libncurses.so.5.6 /lib64/libtinfo.so.5` on 64-bit systems or `ln -s libncurses.so.5.6 /lib/libtinfo.so.5` on 32-bit systems.

Debug versions of the `mysqld` binary are available as `mysqld-debug`. To compile your own debug version of MySQL from a source distribution, use the appropriate configuration options to enable debugging support. See [Installing MySQL from Source](#).

To install and use a MySQL binary distribution, the command sequence looks like this:

```
shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> scripts/mysql_install_db --user=mysql
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

Note

This procedure assumes that you have `root` (administrator) access to your system. Alternatively, you can prefix each command using the `sudo` (Linux) or `pfexec` (Solaris) command.

Note

The procedure does not assign passwords to MySQL accounts. To do so, use the instructions in [Securing the Initial MySQL Accounts](#).

As of MySQL 5.6.8, `mysql_install_db` creates a default option file named `my.cnf` in the base installation directory. This file is created from a template included in the distribution package named `my-default.cnf`. For more information, see [Using a Sample Default Server Configuration File](#).

A more detailed version of the preceding description for installing a binary distribution follows.

Create a mysql User and Group

If your system does not already have a user and group to use for running `mysqld`, you may need to create them. The following commands add the `mysql` group and the `mysql` user. You might want to call the user and group something else instead of `mysql`. If so, substitute the appropriate name in the

following instructions. The syntax for `useradd` and `groupadd` may differ slightly on different versions of Unix/Linux, or they may have different names such as `adduser` and `addgroup`.

```
shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
```

Note

Because the user is required only for ownership purposes, not login purposes, the `useradd` command uses the `-r` and `-s /bin/false` options to create a user that does not have login permissions to your server host. Omit these options if your `useradd` does not support them.

Obtain and Unpack the Distribution

Pick the directory under which you want to unpack the distribution and change location into it. The example here unpacks the distribution under `/usr/local`. The instructions, therefore, assume that you have permission to create files and directories in `/usr/local`. If that directory is protected, you must perform the installation as `root`.

```
shell> cd /usr/local
```

Obtain a distribution file using the instructions in [How to Get MySQL](#). For a given release, binary distributions for all platforms are built from the same MySQL source distribution.

Unpack the distribution, which creates the installation directory. `tar` can uncompress and unpack the distribution if it has `z` option support:

```
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
```

The `tar` command creates a directory named `mysql-VERSION-OS`.

To install MySQL from a compressed `tar` file binary distribution, your system must have GNU `gunzip` to uncompress the distribution and a reasonable `tar` to unpack it. If your `tar` program supports the `z` option, it can both uncompress and unpack the file.

GNU `tar` is known to work. The standard `tar` provided with some operating systems is not able to unpack the long file names in the MySQL distribution. You should download and install GNU `tar`, or if available, use a preinstalled version of GNU `tar`. Usually this is available as `gnutar`, `gtar`, or as `tar` within a GNU or Free Software directory, such as `/usr/sfw/bin` or `/usr/local/bin`. GNU `tar` is available from <http://www.gnu.org/software/tar/>.

If your `tar` does not have `z` option support, use `gunzip` to unpack the distribution and `tar` to unpack it. Replace the preceding `tar` command with the following alternative command to uncompress and extract the distribution:

```
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
```

Next, create a symbolic link to the installation directory created by `tar`:

```
shell> ln -s full-path-to-mysql-VERSION-OS mysql
```

The `ln` command makes a symbolic link to the installation directory. This enables you to refer more easily to it as `/usr/local/mysql`. To avoid having to type the path name of client programs always when you are working with MySQL, you can add the `/usr/local/mysql/bin` directory to your `PATH` variable:

```
shell> export PATH=$PATH:/usr/local/mysql/bin
```

Perform Postinstallation Setup

The remainder of the installation process involves setting distribution ownership and access permissions, initializing the data directory, starting the MySQL server, and setting up the configuration file. For instructions, see [Postinstallation Setup and Testing](#).

Chapter 2 Starting the Server for the First Time on Windows

This section gives a general overview of starting the MySQL server. The following sections provide more specific information for starting the MySQL server from the command line or as a Windows service.

The information here applies primarily if you installed MySQL using the `noinstall` version, or if you wish to configure and test MySQL manually rather than with the GUI tools.

Note

MySQL server will automatically start after using MySQL Installer, and [MySQL Notifier](#) can be used to start/stop/restart at any time.

The examples in these sections assume that MySQL is installed under the default location of `C:\Program Files\MySQL\MySQL Server 5.6`. Adjust the path names shown in the examples if you have MySQL installed in a different location.

Clients have two options. They can use TCP/IP, or they can use a named pipe if the server supports named-pipe connections.

MySQL for Windows also supports shared-memory connections if the server is started with the `--shared-memory` option. Clients can connect through shared memory by using the `--protocol=MEMORY` option.

For information about which server binary to run, see [Selecting a MySQL Server Type](#).

Testing is best done from a command prompt in a console window (or “DOS window”). In this way you can have the server display status messages in the window where they are easy to see. If something is wrong with your configuration, these messages make it easier for you to identify and fix any problems.

To start the server, enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld" --console
```

For a server that includes [InnoDB](#) support, you should see the messages similar to those following as it starts (the path names and sizes may differ):

```
InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25 InnoDB: Started
```

When the server finishes its startup sequence, you should see something like this, which indicates that the server is ready to service client connections:

```
mysqld: ready for connections
Version: '5.6.45' socket: '' port: 3306
```

The server continues to write to the console any further diagnostic output it produces. You can open a new console window in which to run client programs.

If you omit the `--console` option, the server writes diagnostic output to the error log in the data directory (`C:\Program Files\MySQL\MySQL Server 5.6\data` by default). The error log is the file with the `.err` extension, and may be set using the `--log-error` option.

Note

The accounts that are listed in the MySQL grant tables initially have no passwords. After starting the server, you should set up passwords for them using the instructions in [Securing the Initial MySQL Accounts](#).

Chapter 3 MySQL Notifier

Table of Contents

3.1 MySQL Notifier Usage	7
3.2 Setting Up Remote Monitoring in MySQL Notifier	13

MySQL Notifier is a tool that enables you to monitor and adjust the status of your local and remote MySQL server instances through an indicator that resides in the Microsoft Windows taskbar. MySQL Notifier also gives quick access to MySQL Workbench through its context menu.

MySQL Notifier is installed by using MySQL Installer. It can be loaded automatically when Microsoft Windows is started.

To install, download and execute the [MySQL Installer](#). With MySQL Notifier selected from Applications, proceed with the installation. See the [MySQL Installer manual](#) for additional details.

For notes detailing the changes in each release of MySQL Notifier, see the [MySQL Notifier Release Notes](#).

Visit the [MySQL Notifier forum](#) for additional MySQL Notifier help and support.

Features include:

- Start, stop, and restart instances of the MySQL server.
- Automatically detects (and adds) new MySQL server services. These are listed under **Manage Monitored Items**, and may also be configured.
- The Tray icon changes, depending on the status. It is a right-pointing green triangle if all monitored MySQL server instances are running or a red square if at least one service is stopped. The **Update MySQL Notifier tray icon based on service status** option, which dictates this behavior, is enabled by default for each service.
- Links to other applications like MySQL Workbench, MySQL Installer, and the MySQL Utilities. For example, choosing **Manage Instance** will load the MySQL Workbench Server Administration window for that particular instance.
- If MySQL Workbench is also installed, then the **Manage Instance** and **SQL Editor** options are available for local (but not remote) MySQL instances.
- Monitors both local and remote MySQL instances.

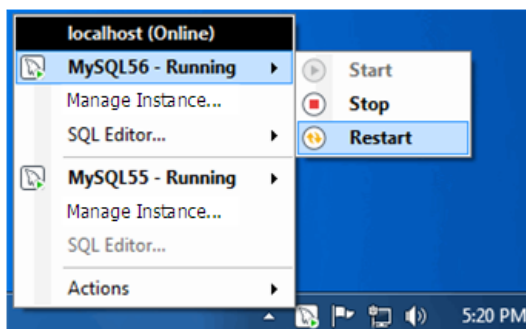
3.1 MySQL Notifier Usage

MySQL Notifier provides visual status information for the MySQL servers that are monitored on both local or remote computers. The MySQL Notifier icon in the taskbar changes color to indicate the current status: Running or Stopped.

MySQL Notifier automatically adds discovered MySQL services on the local computer. By default, the **Automatically add new services whose name contains** option is enabled and set to `mysql`. Related notification options include being notified when new services are either discovered or experience status changes, and are also enabled by default. Uninstalling a service removes the service from MySQL Notifier.

Clicking the MySQL Notifier icon from the Windows taskbar reveals the MySQL Notifier main menu, which lists each MySQL server separately and displays its current status. You can start, stop, or restart each MySQL server from the menu as the following figure shows. When MySQL Workbench is installed locally, the **Manage Instance** and **SQL Editor** menu items start the application.

Figure 3.1 MySQL Notifier Service Instance Menu



The **Actions** menu includes the following items:

- **Manage Monitored Items**
- **Launch MySQL Installer** (Only when the product is installed.)
- **Check for Updates** (Only when MySQL Installer is installed.)
- **MySQL Utilities Shell** (Only when the product is installed.)
- **Refresh Status**
- **Options**
- **About**
- **Close MySQL Notifier**

The main menu does not show the **Actions** menu when there are no services being monitored by MySQL Notifier.

MySQL Notifier Options

The **Actions**, **Options** menu provides a set of options that configure MySQL Notifier operations. Options are grouped into the following categories: **General Options**, **Notification Options**, and **MySQL Server Connections Options**.

Click **Accept** to enable the selected options or **Cancel** to ignore all changes. Click **Reset to Defaults** and then **Accept** to apply default option values.

General Options. This group includes:

- **Use colorful status icons:** Enables a colorful style of icons for the tray of MySQL Notifier. Selected by default.
- **Run at Windows Startup:** Allows the application to be loaded when Microsoft Windows starts. Deselected by default.
- **Automatically Check For Updates Every # Weeks:** Checks for a new version of MySQL Notifier, and runs this check every # weeks. Selected by default with the updates every four weeks.
- **Automatically add new services whose name contains:** The text used to filter services and add them automatically to the monitored list of the local computer running MySQL Notifier and on remote computers already monitoring Windows services. Selected by default for names containing `mysql`.
- **Ping monitored MySQL Server instances every # seconds:** The interval (in seconds) to ping monitored MySQL Server instances for status changes. Longer intervals might be necessary if the list of monitored remote instances is large. 30 seconds by default.

Notification Options. This group includes:

- **Notify me when a service is automatically added:** Display a balloon notification from the taskbar when a newly discovered service is added to the monitored services list. Selected by default.
- **Notify me when a service changes status:** Displays a balloon notification from the taskbar when a monitored service changes its status. Selected by default.

MySQL Server Connections Options. This group includes:

- **Automatic connections migration delayed until:** When there are connections to migrate to MySQL Workbench (if installed), this option postpones the migration by one hour, one day, one week, one month, or indefinitely.

Managing Monitored Items

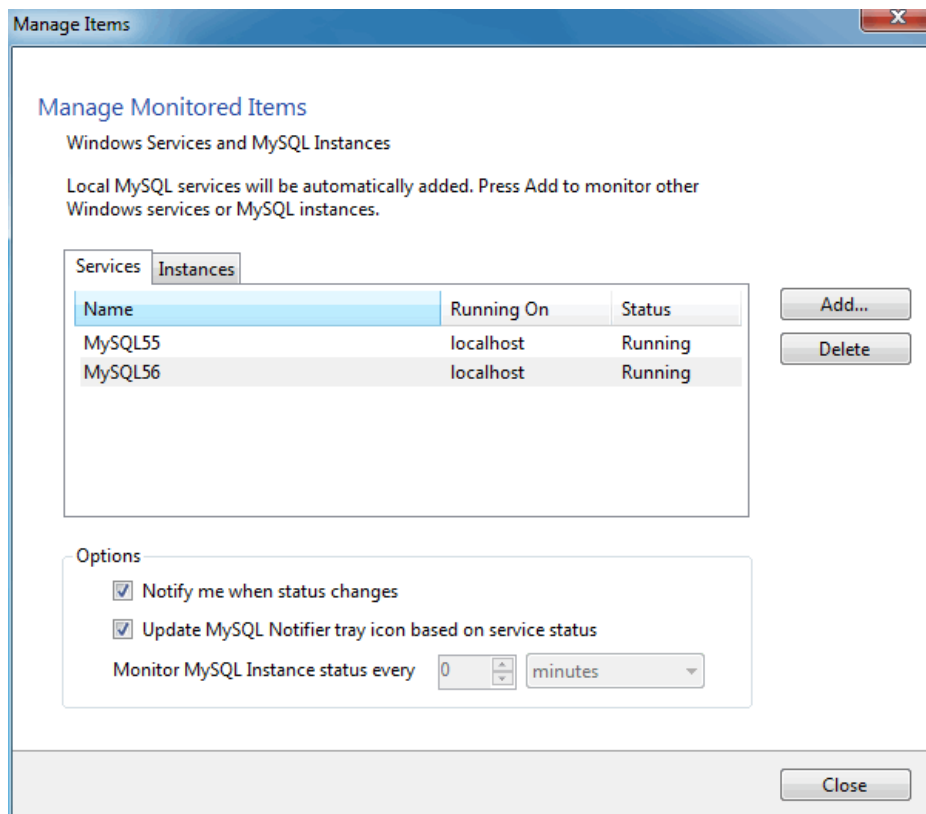
The **Actions, Manage Monitored Items** menu enables you to add, configure, and delete the services and MySQL instances you intend to monitor. The **Manage Items** window has two tabs: **Services** and **Instances**.

Services Tab. When MySQL is configured as a local service, MySQL Notifier adds the service to the **Services** tab automatically. With the **Services** tab open, you can select the following options that apply to all services being monitored:

- **Notify me when status changes**
- **Update MySQL Notifier tray icon based on service status**

The next figure shows the **Services** tab open and both options selected. This tab shows the service name, the computer where the service is hosted, and the current status of the service.

Figure 3.2 MySQL Notifier: Manage Services

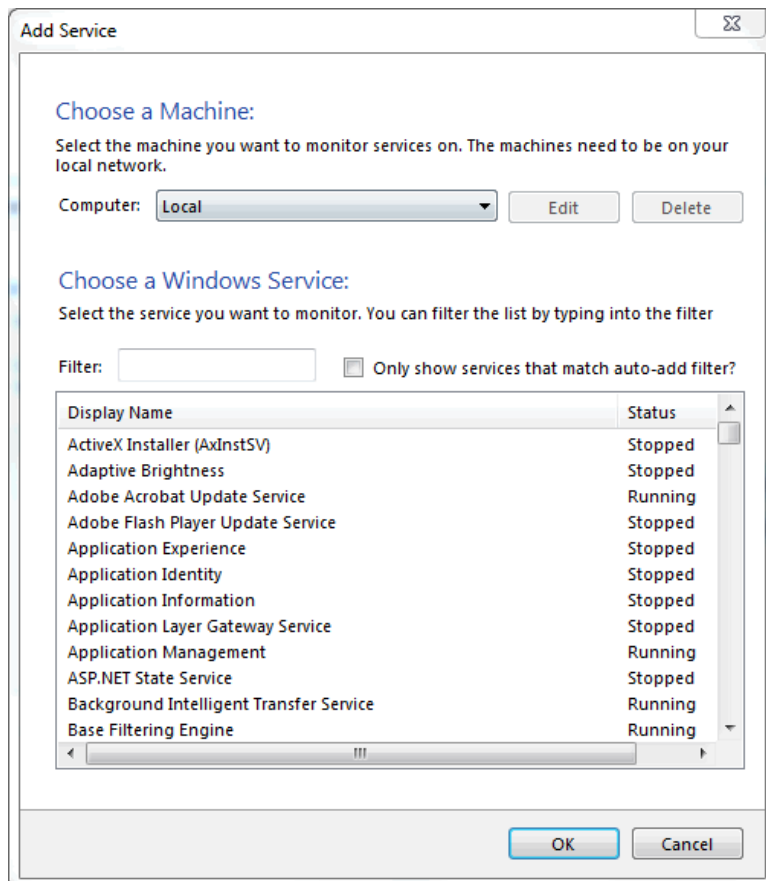


To stop monitoring a service, select it from the list of monitored services and click **Delete**.

Click **Add** and then **Windows Service** to open the **Add Service** window. To add a new service, select a computer from the drop-down list, choose a service from the list, and then click **OK** to accept. Use the **Filter** field to reduce the set of services in the list or select **Only show services that match auto-add filter?** to reuse the general-options filter from the **Options** menu.

A variety of Windows services (including MySQL) may be selected as the following figure shows. In addition to the service name, the list shows the current status of each Windows services for the selected computer.

Figure 3.3 MySQL Notifier: Adding New Services

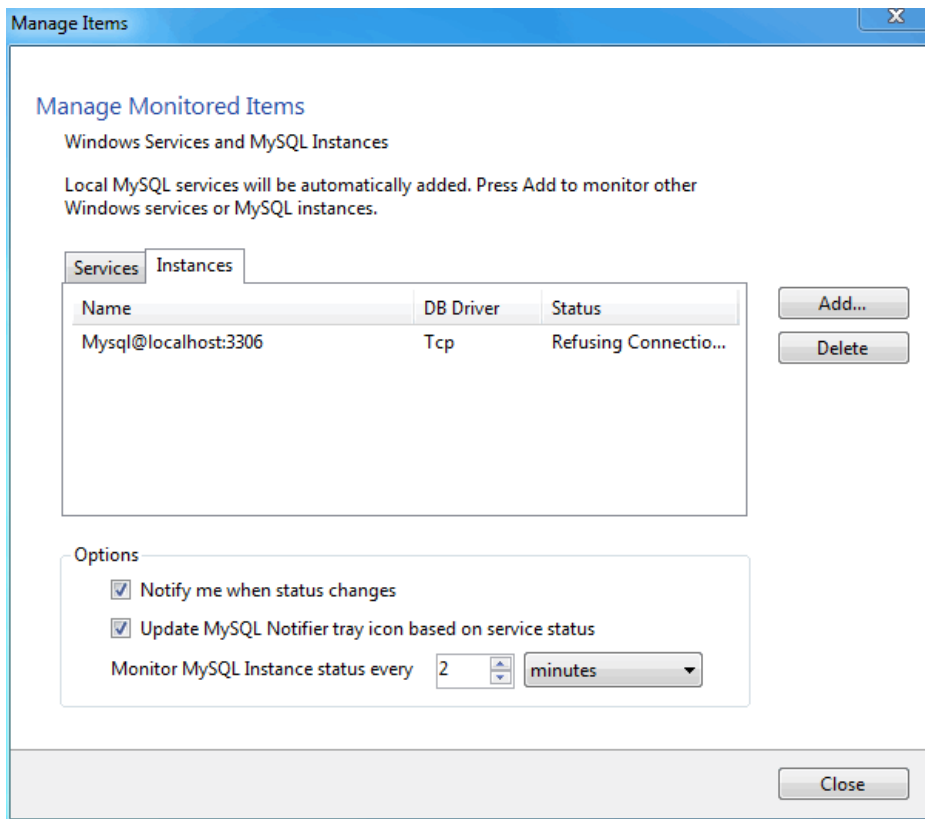


Instances Tab. When MySQL is configured as a MySQL instance, MySQL Notifier adds the instance to the **Instances** tab automatically. With the **Instances** tab open, you can select the following options that apply to each instance being monitored:

- **Notify me when status changes**
- **Update MySQL Notifier tray icon based on service status**
- **Monitor MySQL Instance status every [#] [seconds | minutes | hours | days]**

The next figure shows the **Instances** tab open and both options selected. Monitoring the instance status is set to every two minutes in this example. This tab shows the instance name, the database driver, and the current status of the instance.

Figure 3.4 MySQL Notifier: Manage MySQL Instances

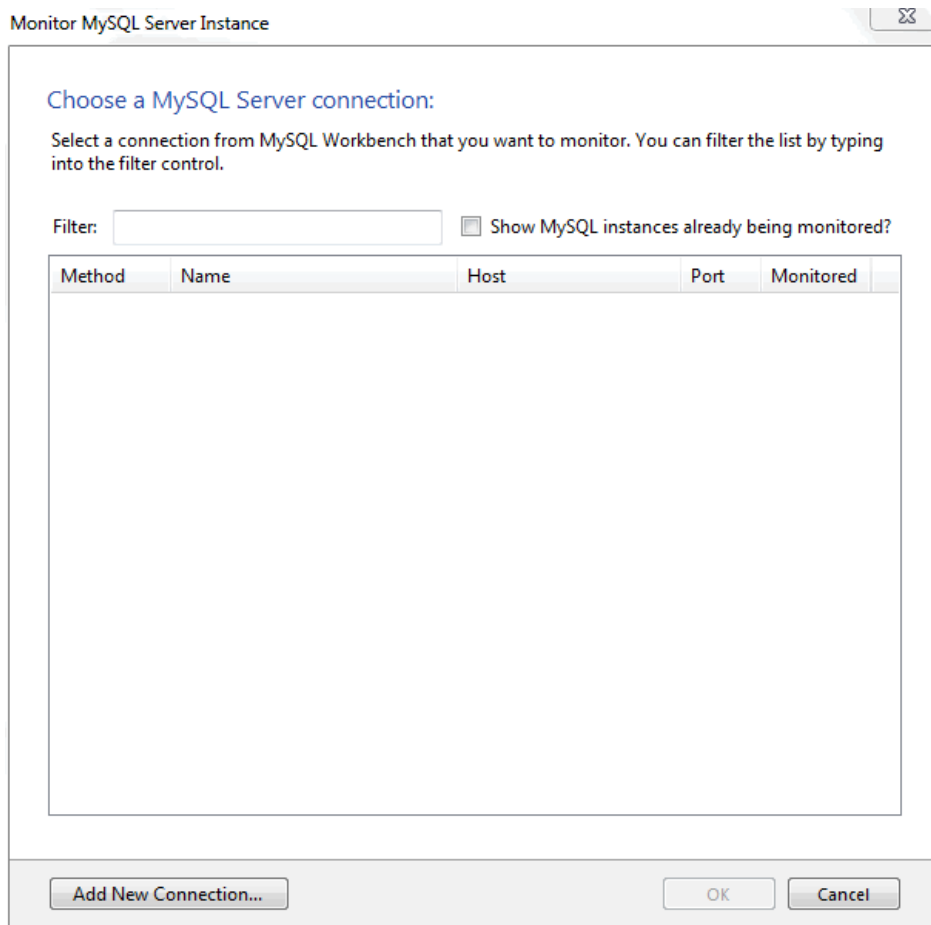


To stop monitoring an instance, select it from the list of monitored MySQL instances and click **Delete**.

Click **Add** and then **MySQL Instances** to open the **Monitor MySQL Server Instance** window. Use the **Filter** field to reduce the set of instances in the list or select **Show MySQL instances already being monitored?** to show monitored items only.

Optionally, you can select a connection from MySQL Workbench to monitor. Click **Add New Connection**, shown in the next figure, to create a new connection.

Figure 3.5 MySQL Notifier: Adding New Instances



Troubleshooting

For issues that are not documented here, visit the [MySQL Notifier Support Forum](#) for MySQL Notifier help and support.

- *Problem:* attempting to start/stop/restart a MySQL service might generate an error similar to "The Service **MySQLVERSION** failed the most recent status change request with the message "The service **mysqlVERSION** was not found in the Windows Services".

Explanation: this is a case-sensitivity issue, in that the service name is **MySQLVERSION** compared to having **mysqlVERSION** in the configuration file.

Solution: either update your MySQL Notifier configuration file with the correct information, or stop MySQL Notifier and delete this configuration file. The MySQL Notifier configuration file is located at `%APPDATA%\Oracle\MySQL Notifier\settings.config` where `%APPDATA%` is a variable and depends on your system. A typical location is "C:\Users\YourUsername\AppData\Roaming\Oracle\MySQL Notifier\settings.config" where *YourUsername* is your system user name. In this file, and within the ServerList section, change the ServerName values from lowercase to the actual service names. For example, change **mysqlVERSION** to **MySQLVERSION**, save, and then restart MySQL Notifier. Alternatively, stop MySQL Notifier, delete this file, then restart MySQL Notifier.

- *Problem:* when connecting to a remote computer for the purpose of monitoring a remote Windows service, the **Add Service** dialog does not always show all the services shown in the Windows Services console.

Explanation: this behavior is governed by the operating system and the outcome is expected when working with nondomain user accounts. For a complete description of the behavior, see the [User Account Control and WMI](#) article from Microsoft.

Solution: when the remote computer is in a compatible domain, it is recommended that domain user accounts are used to connect through WMI to a remote computer. For detailed setup instructions using WMI, see [Section 3.2, “Setting Up Remote Monitoring in MySQL Notifier”](#).

Alternatively, when domain user accounts are not available, Microsoft provides a less secure workaround that should only be implemented with caution. For more information, see the [Description of User Account Control and remote restrictions in Windows Vista](#) KB article from Microsoft.

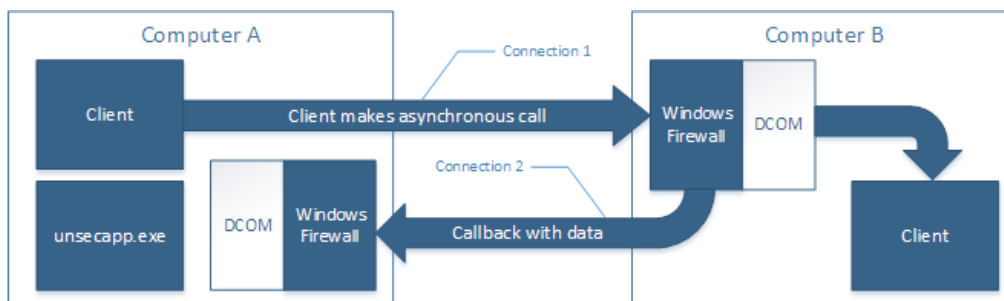
3.2 Setting Up Remote Monitoring in MySQL Notifier

MySQL Notifier uses Windows Management Instrumentation (WMI) to manage and monitor services on remote computers. This section explains how it works and how to set up your system to monitor remote MySQL instances.

In order to configure WMI, it is important to understand that the underlying Distributed Component Object Model (DCOM) architecture is doing the WMI work. Specifically, MySQL Notifier is using asynchronous notification queries on remote Microsoft Windows hosts as .NET events. These events send an asynchronous callback to the computer running MySQL Notifier so it knows when a service status has changed on the remote computer. Asynchronous notifications offer the best performance compared to semisynchronous notifications or synchronous notifications that use timers.

As the following figure shows, asynchronous notification requires the remote computer to send a callback to the client computer (thus opening a reverse connection), so the Windows Firewall and DCOM settings must be properly configured for the communication to function properly. The client (Computer A), which includes an unsecured application (`unsecapp.exe` in this example), makes an asynchronous call to a remote computer (Computer B) and receives a call back with data.

Figure 3.6 MySQL Notifier Distributed Component Object Model (DCOM)



Most of the common errors thrown by asynchronous WMI notifications are related to Windows Firewall blocking the communication, or to DCOM / WMI settings not being set up properly. For a list of common errors with solutions, see [Common Errors](#).

The following steps are required to make WMI function. These steps are divided between two machines. A single host computer that runs MySQL Notifier (Computer A), and multiple remote machines that are being monitored (Computer B).

Computer running MySQL Notifier (Computer A)

1. Enable remote administration by either editing the **Group Policy Editor**, or using `NETSH`:
 - Using the **Group Policy Editor**:
 - a. Click **Start**, click **Run**, type `GPEDIT.MSC`, and then click **OK**.
 - b. Under the **Local Computer Policy** heading, expand **Computer Configuration**.

- c. Expand **Administrative Templates**, then **Network, Network Connections**, and then **Windows Firewall**.
- d. If the computer is in the domain, then double-click **Domain Profile**; otherwise, double-click **Standard Profile**.
- e. Double-click **Windows Firewall: Allow inbound remote administration exception** to open a configuration window.
- f. Check the **Enabled** option button and then click **OK**.

Using the [NETSH](#) command:

Note

The "netsh firewall" command is deprecated as of Microsoft Server 2008 and Vista, and replaced with "netsh advfirewall firewall".

- a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and select **Run as Administrator**).
- b. Execute the following command:

```
NETSH advfirewall firewall set service RemoteAdmin enable
```

2. Open the DCOM port TCP 135:

- a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and select **Run as Administrator**).
- b. Execute the following command:

```
NETSH advfirewall firewall add rule name=DCOM_TCP135 protocol=TCP localport=135 dir=in action=allow
```

3. Add the client application that contains the sink for the callback ([MySQLNotifier.exe](#)) to the Windows Firewall Exceptions List (use either the Windows Firewall configuration or [NETSH](#)):

Using the Windows Firewall configuration:

- a. In the Control Panel, double-click **Windows Firewall**.
- b. In the Windows Firewall window, click **Allow a program or feature through Windows Firewall**.
- c. In the Allowed Programs window, click **Change Settings** and do one of the following:
 - If [MySQLNotifier.exe](#) is in the Allowed programs and features list, make sure it is checked for the type of networks the computer connects to (Private, Public or both).
 - If [MySQLNotifier.exe](#) is not in the list, click **Allow another program**.
 - i. In the **Add a Program** window, select the [MySQLNotifier.exe](#) if it exists in the Programs list, otherwise click **Browse** and go to the directory where [MySQLNotifier.exe](#) was installed to select it, then click **Add**.
 - ii. Make sure [MySQLNotifier.exe](#) is checked for the type of networks the computer connects to (Private, Public or both).

Using the [NETSH](#) command:

- a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and click **Run as Administrator**).
- b. Execute the following command, where you change "*[YOUR_INSTALL_DIRECTORY]*":

```
NETSH advfirewall firewall add rule name=MySQLNotifier program=[YOUR_INSTALL_DIRECTORY]\MySQLNot
```

4. If Computer B is either a member of **WORKGROUP** or is in a different domain that is untrusted by Computer A, then the callback connection (Connection 2) is created as an Anonymous connection. To grant Anonymous connections DCOM Remote Access permissions:
 - a. Click **Start**, click **Run**, type **DCOMCNFG**, and then click **OK**.
 - b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.
 - c. In the My Computer Properties dialog box, click the **COM Security** tab.
 - d. Under Access Permissions, click **Edit Limits**.
 - e. In the Access Permission dialog box, select **ANONYMOUS LOGON name** in the Group or user names box. In the Allow column under Permissions for User, select **Remote Access**, and then click **OK**.

Monitored Remote Computer (Computer B)

If the user account that is logged on to the computer running the MySQL Notifier (Computer A) is a local administrator on the remote computer (Computer B), such that the same account is an administrator on Computer B, you can skip to the "Allow for remote administration" step.

Setting DCOM security to allow a non-administrator user to access a computer remotely:

1. Grant "DCOM remote launch" and activation permissions for a user or group:
 - a. Click **Start**, click **Run**, type **DCOMCNFG**, and then click **OK**.
 - b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.
 - c. In the My Computer Properties dialog box, click the **COM Security** tab.
 - d. Under Launch and Activation Permission, click **Edit Limits**.
 - e. In the **Launch and Activation Permission** dialog box, follow these steps if your name or your group does not appear in the Groups or user names list:
 - i. In the **Launch and Activation Permission** dialog box, click **Add**.
 - ii. In the Select Users or Groups dialog box, add your name and the group in the **Enter the object names to select** box, and then click **OK**.
 - f. In the **Launch and Activation Permission** dialog box, select your user and group in the Group or user names box. In the Allow column under Permissions for User, select **Remote Launch**, select **Remote Activation**, and then click **OK**.

Grant DCOM remote access permissions:

- a. Click **Start**, click **Run**, type **DCOMCNFG**, and then click **OK**.

- b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.
 - c. In the My Computer Properties dialog box, click the **COM Security** tab.
 - d. Under Access Permissions, click **Edit Limits**.
 - e. In the Access Permission dialog box, select **ANONYMOUS LOGON name** in the Group or user names box. In the Allow column under Permissions for User, select **Remote Access**, and then click **OK**.
2. Allowing non-administrator users access to a specific WMI namespace:
- a. In the Control Panel, double-click **Administrative Tools**.
 - b. In the Administrative Tools window, double-click **Computer Management**.
 - c. In the Computer Management window, expand the **Services and Applications** tree.
 - d. Right-click the WMI Control icon and select **Properties**.
 - e. In the WMI Control Properties window, click the **Security** tab.
 - f. In the Security tab, select the namespace and click **Security**. Root/CIMV2 is a commonly used namespace.
 - g. Locate the appropriate account and check **Remote Enable** in the Permissions list.
3. Allow for remote administration by either editing the **Group Policy Editor** or using [NETSH](#):

Using the **Group Policy Editor**:

- a. Click **Start**, click **Run**, type `GPEDIT.MSC`, and then click **OK**.
- b. Under the Local Computer Policy heading, double-click **Computer Configuration**.
- c. Double-click **Administrative Templates**, then **Network**, **Network Connections**, and then **Windows Firewall**.
- d. If the computer is in the domain, then double-click **Domain Profile**; otherwise, double-click **Standard Profile**.
- e. Click **Windows Firewall: Allow inbound remote administration exception**.
- f. On the Action menu either select **Edit**, or double-click the selection from the previous step.
- g. Check the **Enabled** radio button, and then click **OK**.

Using the [NETSH](#) command:

- a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and click **Run as Administrator**).
- b. Execute the following command:

```
NETSH advfirewall firewall set service RemoteAdmin enable
```

4. Confirm that the user account you are logging in with uses the [Name](#) value and not the [Full Name](#) value:
- a. In the **Control Panel**, double-click **Administrative Tools**.

- b. In the **Administrative Tools** window, double-click **Computer Management**.
- c. In the **Computer Management** window, expand the **System Tools** then **Local Users and Groups**.
- d. Click the **Users** node, and on the right side panel locate your user and make sure it uses the **Name** value to connect, and not the **Full Name** value.

Common Errors

- [0x80070005](#)
 - DCOM Security was not configured properly (see Computer B, the [Setting DCOM security...](#) step).
 - The remote computer (Computer B) is a member of WORKGROUP or is in a domain that is untrusted by the client computer (Computer A) (see Computer A, the [Grant Anonymous connections DCOM Remote Access permissions](#) step).
- [0x8007000E](#)
 - The remote computer (Computer B) is a member of WORKGROUP or is in a domain that is untrusted by the client computer (Computer A) (see Computer A, the [Grant Anonymous connections DCOM Remote Access permissions](#) step).
- [0x80041003](#)
 - Access to the remote WMI namespace was not configured properly (see Computer B, the [Allowing non-administrator users access to a specific WMI namespace](#) step).
- [0x800706BA](#)
 - The DCOM port is not open on the client computers (Computer A) firewall. See the [Open the DCOM port TCP 135](#) step for Computer A.
 - The remote computer (Computer B) is inaccessible because its network location is set to Public. Make sure you can access it through the Windows Explorer.

Chapter 4 The Server Shutdown Process

The server shutdown process takes place as follows:

1. The shutdown process is initiated.

This can occur initiated several ways. For example, a user with the [SHUTDOWN](#) privilege can execute a `mysqladmin shutdown` command. `mysqladmin` can be used on any platform supported by MySQL. Other operating system-specific shutdown initiation methods are possible as well: The server shuts down on Unix when it receives a [SIGTERM](#) signal. A server running as a service on Windows shuts down when the services manager tells it to.

2. The server creates a shutdown thread if necessary.

Depending on how shutdown was initiated, the server might create a thread to handle the shutdown process. If shutdown was requested by a client, a shutdown thread is created. If shutdown is the result of receiving a [SIGTERM](#) signal, the signal thread might handle shutdown itself, or it might create a separate thread to do so. If the server tries to create a shutdown thread and cannot (for example, if memory is exhausted), it issues a diagnostic message that appears in the error log:

```
Error: Can't create thread to kill server
```

3. The server stops accepting new connections.

To prevent new activity from being initiated during shutdown, the server stops accepting new client connections by closing the handlers for the network interfaces to which it normally listens for connections: the TCP/IP port, the Unix socket file, the Windows named pipe, and shared memory on Windows.

4. The server terminates current activity.

For each thread associated with a client connection, the server breaks the connection to the client and marks the thread as killed. Threads die when they notice that they are so marked. Threads for idle connections die quickly. Threads that currently are processing statements check their state periodically and take longer to die. For additional information about thread termination, see [KILL Syntax](#), in particular for the instructions about killed [REPAIR TABLE](#) or [OPTIMIZE TABLE](#) operations on [MyISAM](#) tables.

For threads that have an open transaction, the transaction is rolled back.

Note

If a thread is updating a nontransactional table, an operation such as a multiple-row [UPDATE](#) or [INSERT](#) may leave the table partially updated because the operation can terminate before completion.

If the server is a master replication server, it treats threads associated with currently connected slaves like other client threads. That is, each one is marked as killed and exits when it next checks its state.

If the server is a slave replication server, it stops the I/O and SQL threads, if they are active, before marking client threads as killed. The SQL thread is permitted to finish its current statement (to avoid causing replication problems), and then stops. If the SQL thread is in the middle of a transaction at this point, the server waits until the current replication event group (if any) has finished executing, or until the user issues a [KILL QUERY](#) or [KILL CONNECTION](#) statement. See also [STOP SLAVE Syntax](#). Since nontransactional statements cannot be rolled back, in order to guarantee crash-safe replication, only transactional tables should be used.

Note

To guarantee crash safety on the slave, you must run the slave with `--relay-log-recovery` enabled.

See also [Replication Relay and Status Logs](#)).

5. The server shuts down or closes storage engines.

At this stage, the server flushes the table cache and closes all open tables.

Each storage engine performs any actions necessary for tables that it manages. [InnoDB](#) flushes its buffer pool to disk (unless `innodb_fast_shutdown` is 2), writes the current LSN to the tablespace, and terminates its own internal threads. [MyISAM](#) flushes any pending index writes for a table.

6. The server exits.

Chapter 5 MySQL Server and Server-Startup Programs

Table of Contents

5.1 <code>mysqld</code> — The MySQL Server	21
5.2 <code>mysqld_safe</code> — MySQL Server Startup Script	21
5.3 <code>mysql.server</code> — MySQL Server Startup Script	26
5.4 <code>mysqld_multi</code> — Manage Multiple MySQL Servers	28

This section describes `mysqld`, the MySQL server, and several programs that are used to start the server.

5.1 `mysqld` — The MySQL Server

`mysqld`, also known as MySQL Server, is the main program that does most of the work in a MySQL installation. MySQL Server manages access to the MySQL data directory that contains databases and tables. The data directory is also the default location for other information such as log files and status files.

Note

Some installation packages contain a debugging version of the server named `mysqld-debug`. Invoke this version instead of `mysqld` for debugging support, memory allocation checking, and trace file support (see [Creating Trace Files](#)).

When MySQL server starts, it listens for network connections from client programs and manages access to databases on behalf of those clients.

The `mysqld` program has many options that can be specified at startup. For a complete list of options, run this command:

```
shell> mysqld --verbose --help
```

MySQL Server also has a set of system variables that affect its operation as it runs. System variables can be set at server startup, and many of them can be changed at runtime to effect dynamic server reconfiguration. MySQL Server also has a set of status variables that provide information about its operation. You can monitor these status variables to access runtime performance characteristics.

For a full description of MySQL Server command options, system variables, and status variables, see [The MySQL Server](#). For information about installing MySQL and setting up the initial configuration, see [Installing and Upgrading MySQL](#).

5.2 `mysqld_safe` — MySQL Server Startup Script

`mysqld_safe` is the recommended way to start a `mysqld` server on Unix. `mysqld_safe` adds some safety features such as restarting the server when an error occurs and logging runtime information to an error log. A description of error logging is given later in this section.

`mysqld_safe` tries to start an executable named `mysqld`. To override the default behavior and specify explicitly the name of the server you want to run, specify a `--mysqld` or `--mysqld-version` option to `mysqld_safe`. You can also use `--ledir` to indicate the directory where `mysqld_safe` should look for the server.

Many of the options to `mysqld_safe` are the same as the options to `mysqld`. See [Server Command Options](#).

Options unknown to `mysqld_safe` are passed to `mysqld` if they are specified on the command line, but ignored if they are specified in the `[mysqld_safe]` group of an option file. See [Using Option Files](#).

mysqld_safe reads all options from the [mysqld], [server], and [mysqld_safe] sections in option files. For example, if you specify a [mysqld] section like this, mysqld_safe will find and use the --log-error option:

```
[mysqld]
log-error=error.log
```

For backward compatibility, mysqld_safe also reads [safe_mysqld] sections, but to be current you should rename such sections to [mysqld_safe].

mysqld_safe supports the following options. It also reads option files and supports the options for processing them described at [Command-Line Options that Affect Option-File Handling](#).

Table 5.1 mysqld_safe Options

Format	Description
--basedir	Path to MySQL installation directory
--core-file-size	Size of core file that mysqld should be able to create
--datadir	Path to data directory
--defaults-extra-file	Read named option file in addition to usual option files
--defaults-file	Read only named option file
--help	Display help message and exit
--ledir	Path to directory where server is located
--log-error	Write error log to named file
--malloc-lib	Alternative malloc library to use for mysqld
--mysqld	Name of server program to start (in ledir directory)
--mysqld-version	Suffix for server program name
--nice	Use nice program to set server scheduling priority
--no-defaults	Read no option files
--open-files-limit	Number of files that mysqld should be able to open
--pid-file	Path name of server process ID file
--plugin-dir	Directory where plugins are installed
--port	Port number on which to listen for TCP/IP connections
--skip-kill-mysqld	Do not try to kill stray mysqld processes
--skip-syslog	Do not write error messages to syslog; use error log file
--socket	Socket file on which to listen for Unix socket connections
--syslog	Write error messages to syslog
--syslog-tag	Tag suffix for messages written to syslog
--timezone	Set TZ time zone environment variable to named value
--user	Run mysqld as user having name user_name or numeric user ID user_id

- `--help`

Display a help message and exit.

- `--basedir=dir_name`

The path to the MySQL installation directory.

- `--core-file-size=size`

The size of the core file that `mysqld` should be able to create. The option value is passed to `ulimit -c`.

- `--datadir=dir_name`

The path to the data directory.

- `--defaults-extra-file=file_name`

Read this option file in addition to the usual option files. If the file does not exist or is otherwise inaccessible, the server will exit with an error. `file_name` is interpreted relative to the current directory if given as a relative path name rather than a full path name. This must be the first option on the command line if it is used.

For additional information about this and other option-file options, see [Command-Line Options that Affect Option-File Handling](#).

- `--defaults-file=file_name`

Use only the given option file. If the file does not exist or is otherwise inaccessible, the server will exit with an error. `file_name` is interpreted relative to the current directory if given as a relative path name rather than a full path name. This must be the first option on the command line if it is used.

For additional information about this and other option-file options, see [Command-Line Options that Affect Option-File Handling](#).

- `--ledir=dir_name`

If `mysqld_safe` cannot find the server, use this option to indicate the path name to the directory where the server is located.

As of MySQL 5.6.35, this option is accepted only on the command line, not in option files.

- `--log-error=file_name`

Write the error log to the given file. See [The Error Log](#).

- `--malloc-lib=[lib_name]`

The name of the library to use for memory allocation instead of the system `malloc()` library. As of MySQL 5.6.33, the option value must be one of the directories `/usr/lib`, `/usr/lib64`, `/usr/lib/i386-linux-gnu`, or `/usr/lib/x86_64-linux-gnu`. Prior to MySQL 5.6.33, any library can be used by specifying its path name, but there is a shortcut form to enable use of the `tcmalloc` library that is shipped with binary MySQL distributions for Linux in MySQL 5.6. It is possible that the shortcut form will not work under certain configurations, in which case you should specify a path name instead.

Note

As of MySQL 5.6.31, MySQL distributions no longer include a `tcmalloc` library.

The `--malloc-lib` option works by modifying the `LD_PRELOAD` environment value to affect dynamic linking to enable the loader to find the memory-allocation library when `mysqld` runs:

- If the option is not given, or is given without a value (`--malloc-lib=`), `LD_PRELOAD` is not modified and no attempt is made to use `tcmalloc`.
- If the option is given as `--malloc-lib=tcmalloc`, `mysqld_safe` looks for a `tcmalloc` library in `/usr/lib` and then in the MySQL `pkglibdir` location (for example, `/usr/local/mysql/`

`lib` or whatever is appropriate). If `tmalloc` is found, its path name is added to the beginning of the `LD_PRELOAD` value for `mysqld`. If `tcmalloc` is not found, `mysqld_safe` aborts with an error.

- If the option is given as `--malloc-lib=/path/to/some/library`, that full path is added to the beginning of the `LD_PRELOAD` value. If the full path points to a nonexistent or unreadable file, `mysqld_safe` aborts with an error.
- For cases where `mysqld_safe` adds a path name to `LD_PRELOAD`, it adds the path to the beginning of any existing value the variable already has.

Linux users can use the `libtcmalloc_minimal.so` included in binary packages by adding these lines to the `my.cnf` file:

```
[mysqld_safe]
malloc-lib=tcmalloc
```

Those lines also suffice for users on any platform who have installed a `tcmalloc` package in `/usr/lib`. To use a specific `tcmalloc` library, specify its full path name. Example:

```
[mysqld_safe]
malloc-lib=/opt/lib/libtcmalloc_minimal.so
```

- `--mysqld=prog_name`

The name of the server program (in the `ledir` directory) that you want to start. This option is needed if you use the MySQL binary distribution but have the data directory outside of the binary distribution. If `mysqld_safe` cannot find the server, use the `--ledir` option to indicate the path name to the directory where the server is located.

As of MySQL 5.6.33, this option is accepted only on the command line, not in option files.

- `--mysqld-version=suffix`

This option is similar to the `--mysqld` option, but you specify only the suffix for the server program name. The base name is assumed to be `mysqld`. For example, if you use `--mysqld-version=debug`, `mysqld_safe` starts the `mysqld-debug` program in the `ledir` directory. If the argument to `--mysqld-version` is empty, `mysqld_safe` uses `mysqld` in the `ledir` directory.

As of MySQL 5.6.33, this option is accepted only on the command line, not in option files.

- `--nice=priority`

Use the `nice` program to set the server's scheduling priority to the given value.

- `--no-defaults`

Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read. This must be the first option on the command line if it is used.

For additional information about this and other option-file options, see [Command-Line Options that Affect Option-File Handling](#).

- `--open-files-limit=count`

The number of files that `mysqld` should be able to open. The option value is passed to `ulimit -n`.

Note

You must start `mysqld_safe` as `root` for this to function properly.

- `--pid-file=file_name`

The path name that `mysqld` should use for its process ID file.

- `--plugin-dir=dir_name`

The path name of the plugin directory.

- `--port=port_num`

The port number that the server should use when listening for TCP/IP connections. The port number must be 1024 or higher unless the server is started by the `root` operating system user.

- `--skip-kill-mysqld`

Do not try to kill stray `mysqld` processes at startup. This option works only on Linux.

- `--socket=path`

The Unix socket file that the server should use when listening for local connections.

- `--syslog, --skip-syslog`

`--syslog` causes error messages to be sent to `syslog` on systems that support the `logger` program. `--skip-syslog` suppresses the use of `syslog`; messages are written to an error log file.

When `syslog` is used, the `daemon.err` syslog facility/severity is used for all log messages.

`mysqld_safe` ignores `--syslog` if `--log-error` is also given.

- `--syslog-tag=tag`

For logging to `syslog`, messages from `mysqld_safe` and `mysqld` are written with identifiers of `mysqld_safe` and `mysqld`, respectively. To specify a suffix for the identifiers, use `--syslog-tag=tag`, which modifies the identifiers to be `mysqld_safe-tag` and `mysqld-tag`.

- `--timezone=timezone`

Set the `TZ` time zone environment variable to the given option value. Consult your operating system documentation for legal time zone specification formats.

- `--user={user_name|user_id}`

Run the `mysqld` server as the user having the name `user_name` or the numeric user ID `user_id`. (“User” in this context refers to a system login account, not a MySQL user listed in the grant tables.)

If you execute `mysqld_safe` with the `--defaults-file` or `--defaults-extra-file` option to name an option file, the option must be the first one given on the command line or the option file will not be used. For example, this command will not use the named option file:

```
mysql> mysqld_safe --port=port_num --defaults-file=file_name
```

Instead, use the following command:

```
mysql> mysqld_safe --defaults-file=file_name --port=port_num
```

The `mysqld_safe` script is written so that it normally can start a server that was installed from either a source or a binary distribution of MySQL, even though these types of distributions typically install the server in slightly different locations. (See [Installation Layouts](#).) `mysqld_safe` expects one of the following conditions to be true:

- The server and databases can be found relative to the working directory (the directory from which `mysqld_safe` is invoked). For binary distributions, `mysqld_safe` looks under its working directory

for `bin` and `data` directories. For source distributions, it looks for `libexec` and `var` directories. This condition should be met if you execute `mysqld_safe` from your MySQL installation directory (for example, `/usr/local/mysql` for a binary distribution).

- If the server and databases cannot be found relative to the working directory, `mysqld_safe` attempts to locate them by absolute path names. Typical locations are `/usr/local/libexec` and `/usr/local/var`. The actual locations are determined from the values configured into the distribution at the time it was built. They should be correct if MySQL is installed in the location specified at configuration time.

Because `mysqld_safe` tries to find the server and databases relative to its own working directory, you can install a binary distribution of MySQL anywhere, as long as you run `mysqld_safe` from the MySQL installation directory:

```
shell> cd mysql_installation_directory
shell> bin/mysqld_safe &
```

If `mysqld_safe` fails, even when invoked from the MySQL installation directory, specify the `--ledir` and `--datadir` options to indicate the directories in which the server and databases are located on your system.

In MySQL 5.6.5 and later, `mysqld_safe` tries to use the `sleep` and `date` system utilities to determine how many times it has attempted to start this second, and—if these are present and this is greater than 5 times—is forced to wait 1 full second before starting again. This is intended to prevent excessive CPU usage in the event of repeated failures. (Bug #11761530, Bug #54035)

When you use `mysqld_safe` to start `mysqld`, `mysqld_safe` arranges for error (and notice) messages from itself and from `mysqld` to go to the same destination.

There are several `mysqld_safe` options for controlling the destination of these messages:

- `--log-error=file_name`: Write error messages to the named error file.
- `--syslog`: Write error messages to `syslog` on systems that support the `logger` program.
- `--skip-syslog`: Do not write error messages to `syslog`. Messages are written to the default error log file (`host_name.err` in the data directory), or to a named file if the `--log-error` option is given.

If none of these options is given, the default is `--skip-syslog`.

If `--log-error` and `--syslog` are both given, a warning is issued and `--log-error` takes precedence.

When `mysqld_safe` writes a message, notices go to the logging destination (`syslog` or the error log file) and `stdout`. Errors go to the logging destination and `stderr`.

5.3 mysql.server — MySQL Server Startup Script

MySQL distributions on Unix and Unix-like system include a script named `mysql.server`, which starts the MySQL server using `mysqld_safe`. It can be used on systems such as Linux and Solaris that use System V-style run directories to start and stop system services. It is also used by the macOS Startup Item for MySQL.

`mysql.server` is the script name as used within the MySQL source tree. The installed name might be different; for example, `mysqld` or `mysql`. In the following discussion, adjust the name `mysql.server` as appropriate for your system.

To start or stop the server manually using the `mysql.server` script, invoke it from the command line with `start` or `stop` arguments:


```
shell> mysql.server start
shell> mysql.server stop
```

`mysql.server` changes location to the MySQL installation directory, then invokes `mysqld_safe`. To run the server as some specific user, add an appropriate `user` option to the `[mysqld]` group of the global `/etc/my.cnf` option file, as shown later in this section. (It is possible that you must edit `mysql.server` if you've installed a binary distribution of MySQL in a nonstandard location. Modify it to change location into the proper directory before it runs `mysqld_safe`. If you do this, your modified version of `mysql.server` may be overwritten if you upgrade MySQL in the future; make a copy of your edited version that you can reinstall.)

`mysql.server stop` stops the server by sending a signal to it. You can also stop the server manually by executing `mysqladmin shutdown`.

To start and stop MySQL automatically on your server, you must add start and stop commands to the appropriate places in your `/etc/rc*` files:

- If you use the Linux server RPM package (`MySQL-server-VERSION.rpm`), or a native Linux package installation, the `mysql.server` script may be installed in the `/etc/init.d` directory with the name `mysqld` or `mysql`. See [Installing MySQL on Linux Using RPM Packages from Oracle](#), for more information on the Linux RPM packages.
- If you install MySQL from a source distribution or using a binary distribution format that does not install `mysql.server` automatically, you can install the script manually. It can be found in the `support-files` directory under the MySQL installation directory or in a MySQL source tree. Copy the script to the `/etc/init.d` directory with the name `mysql` and make it executable:

```
shell> cp mysql.server /etc/init.d/mysql
shell> chmod +x /etc/init.d/mysql
```

After installing the script, the commands needed to activate it to run at system startup depend on your operating system. On Linux, you can use `chkconfig`:

```
shell> chkconfig --add mysql
```

On some Linux systems, the following command also seems to be necessary to fully enable the `mysql` script:

```
shell> chkconfig --level 345 mysql on
```

- On FreeBSD, startup scripts generally should go in `/usr/local/etc/rc.d/`. Install the `mysql.server` script as `/usr/local/etc/rc.d/mysql.server.sh` to enable automatic startup. The `rc(8)` manual page states that scripts in this directory are executed only if their base name matches the `*.sh` shell file name pattern. Any other files or directories present within the directory are silently ignored.
- As an alternative to the preceding setup, some operating systems also use `/etc/rc.local` or `/etc/init.d/boot.local` to start additional services on startup. To start up MySQL using this method, append a command like the one following to the appropriate startup file:

```
/bin/sh -c 'cd /usr/local/mysql; ./bin/mysqld_safe --user=mysql &'
```

- For other systems, consult your operating system documentation to see how to install startup scripts.

`mysql.server` reads options from the `[mysql.server]` and `[mysqld]` sections of option files. For backward compatibility, it also reads `[mysql_server]` sections, but to be current you should rename such sections to `[mysql.server]`.

You can add options for `mysql.server` in a global `/etc/my.cnf` file. A typical `my.cnf` file might look like this:

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql
[mysql.server]
basedir=/usr/local/mysql
```

The `mysql.server` script supports the options shown in the following table. If specified, they *must* be placed in an option file, not on the command line. `mysql.server` supports only `start` and `stop` as command-line arguments.

Table 5.2 mysql.server Option-File Options

Option Name	Description	Type
<code>basedir</code>	Path to MySQL installation directory	Directory name
<code>datadir</code>	Path to MySQL data directory	Directory name
<code>pid-file</code>	File in which server should write its process ID	File name
<code>service-startup-timeout</code>	How long to wait for server startup	Integer

- `basedir=dir_name`

The path to the MySQL installation directory.

- `datadir=dir_name`

The path to the MySQL data directory.

- `pid-file=file_name`

The path name of the file in which the server should write its process ID. The server creates the file in the data directory unless an absolute path name is given to specify a different directory.

If this option is not given, `mysql.server` uses a default value of `host_name.pid`. The PID file value passed to `mysqld_safe` overrides any value specified in the `[mysqld_safe]` option file group. Because `mysql.server` reads the `[mysqld]` option file group but not the `[mysqld_safe]` group, you can ensure that `mysqld_safe` gets the same value when invoked from `mysql.server` as when invoked manually by putting the same `pid-file` setting in both the `[mysqld_safe]` and `[mysqld]` groups.

- `service-startup-timeout=seconds`

How long in seconds to wait for confirmation of server startup. If the server does not start within this time, `mysql.server` exits with an error. The default value is 900. A value of 0 means not to wait at all for startup. Negative values mean to wait forever (no timeout).

5.4 `mysqld_multi` — Manage Multiple MySQL Servers

`mysqld_multi` is designed to manage several `mysqld` processes that listen for connections on different Unix socket files and TCP/IP ports. It can start or stop servers, or report their current status.

`mysqld_multi` searches for groups named `[mysqldN]` in `my.cnf` (or in the file named by the `--defaults-file` option). `N` can be any positive integer. This number is referred to in the following discussion as the option group number, or *GNR*. Group numbers distinguish option groups from one

another and are used as arguments to `mysql_d_multi` to specify which servers you want to start, stop, or obtain a status report for. Options listed in these groups are the same that you would use in the `[mysqld]` group used for starting `mysqld`. (See, for example, [Starting and Stopping MySQL Automatically](#).) However, when using multiple servers, it is necessary that each one use its own value for options such as the Unix socket file and TCP/IP port number. For more information on which options must be unique per server in a multiple-server environment, see [Running Multiple MySQL Instances on One Machine](#).

To invoke `mysql_d_multi`, use the following syntax:

```
shell> mysql_d_multi [options] {start|stop|reload|report} [GNR[,GNR] ...]
```

`start`, `stop`, `reload` (stop and restart), and `report` indicate which operation to perform. (`reload` is available as of MySQL 5.6.3.) You can perform the designated operation for a single server or multiple servers, depending on the `GNR` list that follows the option name. If there is no list, `mysql_d_multi` performs the operation for all servers in the option file.

Each `GNR` value represents an option group number or range of group numbers. The value should be the number at the end of the group name in the option file. For example, the `GNR` for a group named `[mysqld17]` is 17. To specify a range of numbers, separate the first and last numbers by a dash. The `GNR` value `10-13` represents groups `[mysqld10]` through `[mysqld13]`. Multiple groups or group ranges can be specified on the command line, separated by commas. There must be no whitespace characters (spaces or tabs) in the `GNR` list; anything after a whitespace character is ignored.

This command starts a single server using option group `[mysqld17]`:

```
shell> mysql_d_multi start 17
```

This command stops several servers, using option groups `[mysqld8]` and `[mysqld10]` through `[mysqld13]`:

```
shell> mysql_d_multi stop 8,10-13
```

For an example of how you might set up an option file, use this command:

```
shell> mysql_d_multi --example
```

`mysql_d_multi` searches for option files as follows:

- With `--no-defaults`, no option files are read.
- With `--defaults-file=file_name`, only the named file is read.
- Otherwise, option files in the standard list of locations are read, including any file named by the `--defaults-extra-file=file_name` option, if one is given. (If the option is given multiple times, the last value is used.)

For additional information about these and other option-file options, see [Command-Line Options that Affect Option-File Handling](#).

Option files read are searched for `[mysql_d_multi]` and `[mysqldN]` option groups. The `[mysql_d_multi]` group can be used for options to `mysql_d_multi` itself. `[mysqldN]` groups can be used for options passed to specific `mysqld` instances.

The `[mysqld]` or `[mysqld_safe]` groups can be used for common options read by all instances of `mysqld` or `mysqld_safe`. You can specify a `--defaults-file=file_name` option to use a different configuration file for that instance, in which case the `[mysqld]` or `[mysqld_safe]` groups from that file will be used for that instance.

`mysql_d_multi` supports the following options.

- `--help`

Display a help message and exit.

- `--example`

Display a sample option file.

- `--log=file_name`

Specify the name of the log file. If the file exists, log output is appended to it.

- `--mysqladmin=prog_name`

The `mysqladmin` binary to be used to stop servers.

- `--mysqld=prog_name`

The `mysqld` binary to be used. You can specify `mysqld_safe` as the value for this option. If you use `mysqld_safe` to start the server, you can include the `mysqld` or `ledir` options in the corresponding `[mysqldN]` option group. These options indicate the name of the server that `mysqld_safe` should start and the path name of the directory where the server is located. (See the descriptions for these options in [Section 5.2, “mysqld_safe — MySQL Server Startup Script”](#).)
Example:

```
[mysqld38]
mysqld = mysqld-debug
ledir  = /opt/local/mysql/libexec
```

- `--no-log`

Print log information to `stdout` rather than to the log file. By default, output goes to the log file.

- `--password=password`

The password of the MySQL account to use when invoking `mysqladmin`. The password value is not optional for this option, unlike for other MySQL programs.

- `--silent`

Silent mode; disable warnings.

- `--tcp-ip`

Connect to each MySQL server through the TCP/IP port instead of the Unix socket file. (If a socket file is missing, the server might still be running, but accessible only through the TCP/IP port.) By default, connections are made using the Unix socket file. This option affects `stop` and `report` operations.

- `--user=user_name`

The user name of the MySQL account to use when invoking `mysqladmin`.

- `--verbose`

Be more verbose.

- `--version`

Display version information and exit.

Some notes about `mysqld_multi`:

- **Most important:** Before using `mysqld_multi` be sure that you understand the meanings of the options that are passed to the `mysqld` servers and *why* you would want to have separate `mysqld` processes. Beware of the dangers of using multiple `mysqld` servers with the same data directory. Use separate data directories, unless you *know* what you are doing. Starting multiple servers with the same data directory does *not* give you extra performance in a threaded system. See [Running Multiple MySQL Instances on One Machine](#).

Important

Make sure that the data directory for each server is fully accessible to the Unix account that the specific `mysqld` process is started as. *Do not* use the Unix `root` account for this, unless you *know* what you are doing. See [How to Run MySQL as a Normal User](#).

- Make sure that the MySQL account used for stopping the `mysqld` servers (with the `mysqladmin` program) has the same user name and password for each server. Also, make sure that the account has the `SHUTDOWN` privilege. If the servers that you want to manage have different user names or passwords for the administrative accounts, you might want to create an account on each server that has the same user name and password. For example, you might set up a common `multi_admin` account by executing the following commands for each server:

```
shell> mysql -u root -S /tmp/mysql.sock -p
Enter password:
mysql> CREATE USER 'multi_admin'@'localhost' IDENTIFIED BY 'multipass';
mysql> GRANT SHUTDOWN ON *.* TO 'multi_admin'@'localhost';
```

See [Access Control and Account Management](#). You have to do this for each `mysqld` server. Change the connection parameters appropriately when connecting to each one. The host name part of the account name must permit you to connect as `multi_admin` from the host where you want to run `mysqld_multi`.

- The Unix socket file and the TCP/IP port number must be different for every `mysqld`. (Alternatively, if the host has multiple network addresses, you can set the `bind_address` system variable to cause different servers to listen to different interfaces.)
- The `--pid-file` option is very important if you are using `mysqld_safe` to start `mysqld` (for example, `--mysqld=mysqld_safe`) Every `mysqld` should have its own process ID file. The advantage of using `mysqld_safe` instead of `mysqld` is that `mysqld_safe` monitors its `mysqld` process and restarts it if the process terminates due to a signal sent using `kill -9` or for other reasons, such as a segmentation fault.
- You might want to use the `--user` option for `mysqld`, but to do this you need to run the `mysqld_multi` script as the Unix superuser (`root`). Having the option in the option file doesn't matter; you just get a warning if you are not the superuser and the `mysqld` processes are started under your own Unix account.

The following example shows how you might set up an option file for use with `mysqld_multi`. The order in which the `mysqld` programs are started or stopped depends on the order in which they appear in the option file. Group numbers need not form an unbroken sequence. The first and fifth `[mysqldN]` groups were intentionally omitted from the example to illustrate that you can have “gaps” in the option file. This gives you more flexibility.

```
# This is an example of a my.cnf file for mysqld_multi.
# Usually this file is located in home dir ~/.my.cnf or /etc/my.cnf
[mysqld_multi]
mysqld      = /usr/local/mysql/bin/mysqld_safe
mysqladmin  = /usr/local/mysql/bin/mysqladmin
user        = multi_admin
password    = my_password
[mysqld2]
socket      = /tmp/mysql.sock2
```

```
port = 3307
pid-file = /usr/local/mysql/data2/hostname.pid2
datadir = /usr/local/mysql/data2
language = /usr/local/mysql/share/mysql/english
user = unix_user1
[mysqld3]
mysqld = /path/to/mysqld_safe
ledir = /path/to/mysqld-binary/
mysqladmin = /path/to/mysqladmin
socket = /tmp/mysql.sock3
port = 3308
pid-file = /usr/local/mysql/data3/hostname.pid3
datadir = /usr/local/mysql/data3
language = /usr/local/mysql/share/mysql/swedish
user = unix_user2
[mysqld4]
socket = /tmp/mysql.sock4
port = 3309
pid-file = /usr/local/mysql/data4/hostname.pid4
datadir = /usr/local/mysql/data4
language = /usr/local/mysql/share/mysql/estonia
user = unix_user3
[mysqld6]
socket = /tmp/mysql.sock6
port = 3311
pid-file = /usr/local/mysql/data6/hostname.pid6
datadir = /usr/local/mysql/data6
language = /usr/local/mysql/share/mysql/japanese
user = unix_user4
```

See [Using Option Files](#).