

MySQL Installation Guide

Abstract

This is the MySQL Installation Guide from the MySQL 5.5 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.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the [MySQL Documentation Library](#).

Licensing information—MySQL 5.5. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 5.5, see [this document](#) 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 5.5, see [this document](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL Cluster NDB 7.2. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL Cluster NDB 7.2, see [this document](#) 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 Cluster NDB 7.2, see [this document](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Document generated on: 2017-08-17 (revision: 53505)

Table of Contents

Preface and Legal Notices	v
1 Installing and Upgrading MySQL	1
2 General Installation Guidance	3
2.1 Which MySQL Version and Distribution to Install	3
2.2 How to Get MySQL	4
2.3 Verifying Package Integrity Using MD5 Checksums or GnuPG	5
2.3.1 Verifying the MD5 Checksum	5
2.3.2 Signature Checking Using GnuPG	5
2.3.3 Signature Checking Using Gpg4win for Windows	13
2.3.4 Signature Checking Using RPM	18
2.4 Installation Layouts	19
2.5 Compiler-Specific Build Characteristics	19
3 Installing MySQL on Unix/Linux Using Generic Binaries	21
4 Installing MySQL from Source	25
4.1 MySQL Layout for Source Installation	27
4.2 Installing MySQL Using a Standard Source Distribution	27
4.3 Installing MySQL Using a Development Source Tree	31
4.4 MySQL Source-Configuration Options	33
4.5 Dealing with Problems Compiling MySQL	46
4.6 MySQL Configuration and Third-Party Tools	48
5 Installing MySQL on Microsoft Windows	49
5.1 MySQL Installation Layout on Microsoft Windows	52
5.2 Choosing An Installation Package	53
5.3 MySQL Installer for Windows	54
5.3.1 MySQL Installer Initial Setup	55
5.3.2 Installation Workflow	59
5.3.3 MySQL Installer Product Catalog and Dashboard	66
5.3.4 MySQLInstallerConsole Reference	69
5.4 MySQL Notifier	73
5.4.1 MySQL Notifier Usage	73
5.4.2 Setting Up Remote Monitoring in MySQL Notifier	80
5.5 Installing MySQL on Microsoft Windows Using an MSI Package	85
5.5.1 Using the MySQL Installation Wizard	87
5.5.2 Automating MySQL Installation on Microsoft Windows Using the MSI Package	90
5.5.3 Removing MySQL When Installed from the MSI Package	91
5.6 MySQL Server Instance Configuration Wizard	92
5.6.1 Starting the MySQL Server Instance Configuration Wizard	92
5.6.2 Choosing a Maintenance Option	93
5.6.3 Choosing a Configuration Type	93
5.6.4 The Server Type Dialog	94
5.6.5 The Database Usage Dialog	96
5.6.6 The InnoDB Tablespace Dialog	97
5.6.7 The Concurrent Connections Dialog	98
5.6.8 The Networking and Strict Mode Options Dialog	99
5.6.9 The Character Set Dialog	100
5.6.10 The Service Options Dialog	100
5.6.11 The Security Options Dialog	101
5.6.12 The Confirmation Dialog	103
5.6.13 MySQL Server Instance Config Wizard: Creating an Instance from the Command Line	104
5.7 Installing MySQL on Microsoft Windows Using a noinstall Zip Archive	107

5.7.1	Extracting the Install Archive	107
5.7.2	Creating an Option File	108
5.7.3	Selecting a MySQL Server Type	109
5.7.4	Starting the Server for the First Time	109
5.7.5	Starting MySQL from the Windows Command Line	111
5.7.6	Customizing the PATH for MySQL Tools	111
5.7.7	Starting MySQL as a Windows Service	112
5.7.8	Testing The MySQL Installation	115
5.8	Troubleshooting a Microsoft Windows MySQL Server Installation	116
5.9	Windows Postinstallation Procedures	117
5.10	Upgrading MySQL on Windows	119
6	Installing MySQL on OS X	123
6.1	General Notes on Installing MySQL on OS X	123
6.2	Installing MySQL on OS X Using Native Packages	124
6.3	Installing a MySQL Launch Daemon	130
6.4	Installing and Using the MySQL Preference Pane	133
7	Installing MySQL on Linux	139
7.1	Installing MySQL on Linux Using RPM Packages	139
7.2	Installing MySQL on Linux Using Debian Packages	144
7.3	Installing MySQL on Linux Using Native Package Managers	145
8	Installing MySQL on Solaris	149
8.1	Installing MySQL on Solaris Using a Solaris PKG	150
9	Postinstallation Setup and Testing	151
9.1	Initializing the Data Directory	151
9.1.1	Problems Running mysql_install_db	153
9.2	Starting the Server	155
9.2.1	Troubleshooting Problems Starting the MySQL Server	156
9.3	Testing the Server	158
9.4	Securing the Initial MySQL Accounts	160
9.5	Starting and Stopping MySQL Automatically	164
10	Upgrading or Downgrading MySQL	167
10.1	Upgrading MySQL	167
10.1.1	Changes Affecting Upgrades to MySQL 5.5	171
10.2	Downgrading MySQL	177
10.2.1	Changes Affecting Downgrades from MySQL 5.5	181
10.2.2	Downgrading from MySQL Enterprise Edition to MySQL Community Server	181
10.3	Rebuilding or Repairing Tables or Indexes	182
10.4	Copying MySQL Databases to Another Machine	184
11	MySQL Program Environment Variables	187
12	Perl Installation Notes	189
12.1	Installing Perl on Unix	189
12.2	Installing ActiveState Perl on Windows	190
12.3	Problems Using the Perl DBI/DBD Interface	191

Preface and Legal Notices

This is the MySQL Installation Guide from the MySQL 5.5 Reference Manual.

Legal Notices

Copyright © 1997, 2017, 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.

Chapter 1 Installing and Upgrading MySQL

This chapter describes how to obtain and install MySQL. A summary of the procedure follows and later sections provide the details. If you plan to upgrade an existing version of MySQL to a newer version rather than install MySQL for the first time, see [Section 10.1, “Upgrading MySQL”](#), for information about upgrade procedures and about issues that you should consider before upgrading.

If you are interested in migrating to MySQL from another database system, see [MySQL 5.5 FAQ: Migration](#), which contains answers to some common questions concerning migration issues.

If you are migrating from MySQL Enterprise Edition to MySQL Community Server, see [Section 10.2.2, “Downgrading from MySQL Enterprise Edition to MySQL Community Server”](#).

Installation of MySQL generally follows the steps outlined here:

- 1. Determine whether MySQL runs and is supported on your platform.**

Please note that not all platforms are equally suitable for running MySQL, and that not all platforms on which MySQL is known to run are officially supported by Oracle Corporation. For information about those platforms that are officially supported, see <http://www.mysql.com/support/supportedplatforms/database.html> on the MySQL Web site.

- 2. Choose which distribution to install.**

Several versions of MySQL are available, and most are available in several distribution formats. You can choose from pre-packaged distributions containing binary (precompiled) programs or source code. When in doubt, use a binary distribution. Oracle also provides access to the MySQL source code for those who want to see recent developments and test new code. To determine which version and type of distribution you should use, see [Section 2.1, “Which MySQL Version and Distribution to Install”](#).

- 3. Download the distribution that you want to install.**

For instructions, see [Section 2.2, “How to Get MySQL”](#). To verify the integrity of the distribution, use the instructions in [Section 2.3, “Verifying Package Integrity Using MD5 Checksums or GnuPG”](#).

- 4. Install the distribution.**

To install MySQL from a binary distribution, use the instructions in [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#).

To install MySQL from a source distribution or from the current development source tree, use the instructions in [Chapter 4, *Installing MySQL from Source*](#).

- 5. Perform any necessary postinstallation setup.**

After installing MySQL, see [Chapter 9, *Postinstallation Setup and Testing*](#) for information about making sure the MySQL server is working properly. Also refer to the information provided in [Section 9.4, “Securing the Initial MySQL Accounts”](#). This section describes how to secure the initial MySQL user accounts, *which have no passwords* until you assign passwords. The section applies whether you install MySQL using a binary or source distribution.

- 6. If you want to run the MySQL benchmark scripts, Perl support for MySQL must be available. See [Chapter 12, *Perl Installation Notes*](#).**

Instructions for installing MySQL on different platforms and environments is available on a platform by platform basis:

- **Unix, Linux, FreeBSD**

For instructions on installing MySQL on most Linux and Unix platforms using a generic binary (for example, a `.tar.gz` package), see [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#).

For information on building MySQL entirely from the source code distributions or the source code repositories, see [Chapter 4, *Installing MySQL from Source*](#)

For specific platform help on installation, configuration, and building from source see the corresponding platform section:

- Linux, including notes on distribution specific methods, see [Chapter 7, *Installing MySQL on Linux*](#).
- Solaris, including PKG and IPS formats, see [Chapter 8, *Installing MySQL on Solaris*](#).
- IBM AIX, see [Chapter 8, *Installing MySQL on Solaris*](#).
- FreeBSD, see [Installing MySQL on FreeBSD](#).

- **Microsoft Windows**

For instructions on installing MySQL on Microsoft Windows, using either the MySQL Installer standalone MSI, or Zipped binary, see [Chapter 5, *Installing MySQL on Microsoft Windows*](#).

For information about managing MySQL instances, see [Section 5.4, “MySQL Notifier”](#).

For details and instructions on building MySQL from source code using Microsoft Visual Studio, see [Chapter 4, *Installing MySQL from Source*](#).

- **OS X**

For installation on OS X, including using both the binary package and native PKG formats, see [Chapter 6, *Installing MySQL on OS X*](#).

For information on making use of an OS X Launch Daemon to automatically start and stop MySQL, see [Section 6.3, “Installing a MySQL Launch Daemon”](#).

For information on the MySQL Preference Pane, see [Section 6.4, “Installing and Using the MySQL Preference Pane”](#).

Chapter 2 General Installation Guidance

Table of Contents

2.1 Which MySQL Version and Distribution to Install	3
2.2 How to Get MySQL	4
2.3 Verifying Package Integrity Using MD5 Checksums or GnuPG	5
2.3.1 Verifying the MD5 Checksum	5
2.3.2 Signature Checking Using GnuPG	5
2.3.3 Signature Checking Using Gpg4win for Windows	13
2.3.4 Signature Checking Using RPM	18
2.4 Installation Layouts	19
2.5 Compiler-Specific Build Characteristics	19

The immediately following sections contain the information necessary to choose, download, and verify your distribution. The instructions in later sections of the chapter describe how to install the distribution that you choose. For binary distributions, see the instructions at [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#) or the corresponding section for your platform if available. To build MySQL from source, use the instructions in [Chapter 4, *Installing MySQL from Source*](#).

2.1 Which MySQL Version and Distribution to Install

MySQL is available on a number of operating systems and platforms. For information about those platforms that are officially supported, see <http://www.mysql.com/support/supportedplatforms/database.html> on the MySQL Web site.

When preparing to install MySQL, decide which version and distribution format (binary or source) to use.

First, decide whether to install a development release or a General Availability (GA) release. Development releases have the newest features, but are not recommended for production use. GA releases, also called production or stable releases, are meant for production use. We recommend using the most recent GA release.

The naming scheme in MySQL 5.5 uses release names that consist of three numbers and an optional suffix; for example, **mysql-5.5.1-m2**. The numbers within the release name are interpreted as follows:

- The first number (**5**) is the major version number.
- The second number (**5**) is the minor version number. Taken together, the major and minor numbers constitute the release series number. The series number describes the stable feature set.
- The third number (**1**) is the version number within the release series. This is incremented for each new bugfix release. In most cases, the most recent version within a series is the best choice.

Release names can also include a suffix to indicate the stability level of the release. Releases within a series progress through a set of suffixes to indicate how the stability level improves. The possible suffixes are:

- **mN** (for example, **m1**, **m2**, **m3**, ...) indicates a milestone number. MySQL development uses a milestone model, in which each milestone introduces a small subset of thoroughly tested features. Following the releases for one milestone, development proceeds with another small number of releases that focuses

on the next set of features. From one milestone to the next, feature interfaces may change or features may even be removed, based on feedback provided by community members who try these early releases. Features within milestone releases may be considered to be of pre-production quality.

- **rc** indicates a Release Candidate (RC). Release candidates are believed to be stable, having passed all of MySQL's internal testing. New features may still be introduced in RC releases, but the focus shifts to fixing bugs to stabilize features introduced earlier within the series.
- Absence of a suffix indicates a General Availability (GA) or Production release. GA releases are stable, having successfully passed through the earlier release stages, and are believed to be reliable, free of serious bugs, and suitable for use in production systems.

Development within a series begins with milestone releases, followed by RC releases, and finally reaches GA status releases.

After choosing which MySQL version to install, decide which distribution format to install for your operating system. For most use cases, a binary distribution is the right choice. Binary distributions are available in native format for many platforms, such as RPM packages for Linux or DMG packages for OS X. Distributions are also available in more generic formats such as Zip archives or compressed `tar` files. On Windows, you can use [the MySQL Installer](#) to install a binary distribution.

Under some circumstances, it may be preferable to install MySQL from a source distribution:

- You want to install MySQL at some explicit location. The standard binary distributions are ready to run at any installation location, but you might require even more flexibility to place MySQL components where you want.
- You want to configure `mysqld` with features that might not be included in the standard binary distributions. Here is a list of the most common extra options used to ensure feature availability:
 - `-DWITH_LIBWRAP=1` for TCP wrappers support.
 - `-DWITH_ZLIB={system|bundled}` for features that depend on compression
 - `-DWITH_DEBUG=1` for debugging support

For additional information, see [Section 4.4, "MySQL Source-Configuration Options"](#).

- You want to configure `mysqld` without some features that are included in the standard binary distributions. For example, distributions normally are compiled with support for all character sets. If you want a smaller MySQL server, you can recompile it with support for only the character sets you need.
- You want to read or modify the C and C++ code that makes up MySQL. For this purpose, obtain a source distribution.
- Source distributions contain more tests and examples than binary distributions.

2.2 How to Get MySQL

Check our downloads page at <http://dev.mysql.com/downloads/> for information about the current version of MySQL and for downloading instructions. For a complete up-to-date list of MySQL download mirror sites, see <http://dev.mysql.com/downloads/mirrors.html>. You can also find information there about becoming a MySQL mirror site and how to report a bad or out-of-date mirror.

To obtain the latest development source, see [Section 4.3, "Installing MySQL Using a Development Source Tree"](#).

2.3 Verifying Package Integrity Using MD5 Checksums or GnuPG

After downloading the MySQL package that suits your needs and before attempting to install it, make sure that it is intact and has not been tampered with. There are three means of integrity checking:

- MD5 checksums
- Cryptographic signatures using [GnuPG](#), the GNU Privacy Guard
- For RPM packages, the built-in RPM integrity verification mechanism

The following sections describe how to use these methods.

If you notice that the MD5 checksum or GPG signatures do not match, first try to download the respective package one more time, perhaps from another mirror site.

2.3.1 Verifying the MD5 Checksum

After you have downloaded a MySQL package, you should make sure that its MD5 checksum matches the one provided on the MySQL download pages. Each package has an individual checksum that you can verify against the package that you downloaded. The correct MD5 checksum is listed on the downloads page for each MySQL product, and you will compare it against the MD5 checksum of the file (product) that you download.

Each operating system and setup offers its own version of tools for checking the MD5 checksum. Typically the command is named `md5sum`, or it may be named `md5`, and some operating systems do not ship it at all. On Linux, it is part of the **GNU Text Utilities** package, which is available for a wide range of platforms. You can also download the source code from <http://www.gnu.org/software/textutils/>. If you have OpenSSL installed, you can use the command `openssl md5 package_name` instead. A Windows implementation of the `md5` command line utility is available from <http://www.fourmilab.ch/md5/>. `winMd5Sum` is a graphical MD5 checking tool that can be obtained from <http://www.nullriver.com/index/products/winmd5sum>. Our Microsoft Windows examples will assume the name `md5.exe`.

Linux and Microsoft Windows examples:

```
shell> md5sum mysql-standard-5.5.58-linux-i686.tar.gz
aaab65abbec64d5e907dcd41b8699945  mysql-standard-5.5.58-linux-i686.tar.gz
```

```
shell> md5.exe mysql-installer-community-5.5.58.msi
aaab65abbec64d5e907dcd41b8699945  mysql-installer-community-5.5.58.msi
```

You should verify that the resulting checksum (the string of hexadecimal digits) matches the one displayed on the download page immediately below the respective package.

Note

Make sure to verify the checksum of the *archive file* (for example, the `.zip`, `.tar.gz`, or `.msi` file) and not of the files that are contained inside of the archive. In other words, verify the file before extracting its contents.

2.3.2 Signature Checking Using GnuPG

Another method of verifying the integrity and authenticity of a package is to use cryptographic signatures. This is more reliable than using [MD5 checksums](#), but requires more work.

We sign MySQL downloadable packages with [GnuPG](#) (GNU Privacy Guard). [GnuPG](#) is an Open Source alternative to the well-known Pretty Good Privacy ([PGP](#)) by Phil Zimmermann. Most Linux distributions ship

with [GnuPG](#) installed by default. Otherwise, see <http://www.gnupg.org/> for more information about [GnuPG](#) and how to obtain and install it.

To verify the signature for a specific package, you first need to obtain a copy of our public GPG build key, which you can download from <http://pgp.mit.edu/>. The key that you want to obtain is named mysq1-build@oss.oracle.com. Alternatively, you can copy and paste the key directly from the following text:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.5 (GNU/Linux)
mQGIBD4+owwRBAC14GIFuFcyEDSIePvEW3SAFUdJbtoQHH/nJKZyQT7h9bPlUWC3
RODjQReyCITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPKbDcK96+OmSLN9brZ
fw2vOUgCmYv2hW0hyDHuvYlQA/BThQoADgJ8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3
BqOxRznNCRcRxAuAuVztHRcEAJooQK1+iSiunZMYDlWufeXfshc57S/+yeJkegNW
hxwR9pRWVArNYJdDRT+rf2RUE3vpquKNQU/hnEIHJRQqYHo8gTxvxXNQC7fJYLV
K2HtkrPbP72vwsEKMYhhr0eKcbtLGfls9krjJ6sBgACyP/Vb7hiPwxh6rDZ7ITnE
kYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgtobZX9qnrAXw+uNDI
QJEXM6F5bi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPANWqcyefep
rv0sxxqUEMcm3o7wwgfN83P0kDasDbs3pjwPhxvzhz6//62zQJ7Q2TXlTUUwgUmVs
ZWFzZSBFbmdpbmVlcm1uZyA8bXlzcWwtYnVpbGRAb3NzLm9yYWNsZS5jb20+iGwE
ExECACwCGyMCHgECF4ACGQEGCwkIBwMCBhUKCQgCAwUWAgMBAAUCWkCFIAUJHirJ
FAAKCRCMcY07UHLh9vCFAJ46pUyVd8BZ2r5CpPMCltmYQ3ceRgCfVPwVsiS0VER
5WUqtAQDt+DoetCIAQQTEQIAKQIbIwYLCQgHAWIEFQIIAWQWAgMBAh4BAheAAhKB
BQJTAdRmBQkaZsvLAAoJEIxxjTtQcuHlX4MAoKNLWAbCBUj96637kv6Xa/fJuX5m
AJwPtmgDfjUe2iuhXdtRFEPT19SB6ohmBBMRAGAmAhsjBgsJCAcDagQVAggDBBYC
AwECHgECF4AFak53PioFCRP7AhUACgkQjHGNO1By4fUmzACeJdfggc9gWTUhgmcM
AomG4RjwuxcAoKfM+U8yMOGELi+TRif7MtKEms6piGkEEEXCACkCGyMGcwkIBwMC
BBUCCAMEFgIDAQIEAQIXgAIZAQUCUZSR0gUJFTchqgAKCRCMcY07UHLh9YtAAJ9X
rA/ymlmoZpZn+A9l8/uwMcTsQCfaQMNq1dNkhH2kyByc3Rx9/W2xfqJARwEEAEAC
AAYFALA96+UACgkQ8aIC+GoXhivrwWf/dtLk/x+NC2VMDlg+vOeM0ggG1lhxZfi
NsEisvvGaz4m8fSFRGe+lbvfvDoKRhxiGXU48RusjixzvBb6KTMuY6JpOVfz9Dj3
H9spYriHa+i6rYySXZlpOhfLiMnTy7NH20vYCyNzSS/ciIUACIfh/2NH8zNT5CNF
luPNRs7HsHzzz7p0LTjtTWiF4cq/IjZ6ZCNrmdj+SiMvjYN9u6sdEKGtoNtpycgD
5HGKR+I7Nd/7v56yhaUe4FpuvsNXig86K9tI6MUFS8CUyy7Hj3kVBZOUWVBM053k
nGdALSygQR50DA3jMgKv14ZnHje2RVRWmFTr5YwOrTmxUSQPLPbNlIkBHAQQAIA
BgUCU1B+vQAKCRAohbcD0zcc8dWwCACWXXWDXIcAWRUw+j3ph8dr9u3SItlJn3wB
c7clpcLkWPuLvTz7lGgzlVB0s8hH4xgkSA+zLz6u56mpUzskF17f1I3Ac9GGpM4
0M5vmmR9hwlDlHdZtGfbd+wkjlqgitNLoRcGdRf/+U7x09GhSS7Bf339sunIX6sM
gXSC4L32D3zDjF5icGdb0kj+3lCrRmp853dGyA3ff9yUiBkxckNawpi7Vz3D2ddU
pOF3BP+8NKpG4P2+srKgkFbd4HidcISQct3rY4vaTkEkLKgOnNA6U4r0YgOa7wIT
SsxFlntMMzArG53QtK0+YkH0KuZR3GY8B7pi+tlgycyVR7mIFo7riQEcbBABAAG
BQJWgVd0AAoJEEZu4b/gk4UKk9MH/Rnt7EccPjSJC5CrB2AU5LY2Dsr+PePI2ubP
WsEdG82qSjJgpbhIH8LSg/PzQoGHiFWMmZwJkrt+dcgLbs3b2VwCNawCE8jOHd
UkQhEowgomdNvHiBHKHjP4/1F68KOPio/2mxYkmpM7Bwf3kB57DJ5CTi3/JLON7
zF40qIs/p09ePvnStpglbbtUn7XPO+1/Ee8VHzimABom52PkIuxNiVUZLVn3Bs
Wqrd5ecuqLk6yzjPxd2XhDHC9Twp168GePru6EzQtusi0m6S/sHgEXqh/IxrfZV
Jl1jF75JvosZq5zeulr0i6k0ij+Ylp6MFffihITZ1gTmk+CLvK2JASIEEAECAAwF
Ak53QS4FAwAsDQAACgkQlxc4m8pXrXwJ8Qf/be/U09mqfoc2sMyhwMpn4/fdBWwf
LkA12FXQDOQmVwH9HsmEjnfUgYKXschZri+DuHXelP718G2aQLubhBsQf9ejKvRF
TzuWMQkdIq+6Koulxv6ofkCev3dlxt02W7nb5yxcPVBPrRFfGfGebJvZa58DymCNg
yGtAU6AOz4veavNmI2+GIDQsY66+tYDvZ+CxwzdYu+HDV9HmrJfc6deM0mnBn7SR
jqzXJPgoTQhithTav6q/R5/2p5NvQ/H84OgS6GjosfGc2duUDzCP/kheMRkfzuyKC
OHQPtJuIj8++gfpHtEU7IDUXlSo3c9n0PdpeBvclsDbpRnCNxQUW4mBot4kBIgQQ
AQIADAUCToi2GQUDABJ1AAAKCRCXELibylefLZAB/9oRqx+NC98UQD/wlxCrytZ
vi/MuPnbgQUPLHEapl0tveI33S/H/xDR/tcGofY4cjAvo5skZXXEWq93Av7PACUb
zkg0X0eSr2oL6wy66xfov72AwSuX+iUK68qtKaLqRLitM02y8aNRV/ggKvt7UMvG
mOvs5yLaYlobyvGaFC2Clfkn0t2M1VnQZCmnYBCwOktPGkExiu2yZMifcYgXqcpH
KVFG59KeF2cM2d4xYM8HJqkSGGW306LFVSyRwG+wbtgLPd5bM/T2b3fF/J35ra
CSMLZearRTq8aygPl+XM7MM2er946aw6jmOsgNBERbvvdQj6LudAZj+8imcXV2K
iQIiBBABAgAMBQjOmdnRBQMAEnUAAoJEJcQujvKV618AvIAlAEFLZJ+Ry7WodKF
5oeQ/ynaYUigzN92fW/9zB8yuQlNgkFJGidYMBciltRlsiziIVJFusr3ZonqAPGK
/SUta9Y6KWLhmc7c5UnEHklq/NfdMZ2WVSiykXlctqw0sbb+zlecEd4G8u9j5i1l
MO1B36rQayYAPoeXL8dY4VyFLVGAQ00rWQBYFZrpw16ATwBWGJP332NSfCk4zZq
6kXEW07q0st3YBgAAGdNqyEzCa4d4pBRsX6189Kjg6GdNcaioF6HO6PLr9fRlL
r5ObCgU+G9gEhfiVwDEV9E+7/Bq2pYZ9whhkBqWQzdpXTNTM24uaEhE01EP05zeC
O214q6mJASIEEAECAAwFAk6rpgEFAwAsDQAACgkQlxc4m8pXrXzAhwf/f9099z16
```

Signature Checking Using GnuPG

3Y5FZVIxexyqXQ/Mct9uKHuXEvnRFYbA49dQLD4S73N+zN7gn9jFeQcBo4w8qVUV
94U/ta/VbLkdNREYp1PM4XY8YE5Wfd9bfyg3q1PbEiVjk995sBF+2+To99YYKst
gXPqj1h0jUfEyDmexOj+hsp8Rc63kvkIx36VBa4ONRYFefGAhKDMigL2YAhc1UkG
tkGTuLm1CGwIV61v1dZD3Rjf5375VFnaHv7eXfWQxCwE+BxG3CURrjfxjaxMTmMP
yAG2rhDp5oTUEvqDYnbko5UxYomrSjvF4FzXwqerElXJUkUzSh0pp7RxBH/1lCx
s7D1Flh1gFQuNlKBiGQQAQIADAUCTrzZHAUDABJ1AAAKCRCXELibyletFMUPB/4s
07dREULIBnAlD6qr3fHsQJNZqbAuyDlvgGGLWzoyEDs+1JMFFlaa+EeLIo1386GU
2DammDC23p3IB79uQhJeD2Z1TcVg4cA64SfF/CHca5coerSrdAiudzU/cgLGtXIP
/OaFamXgdMxAh1oLFBShPCZkyb0OphVa8+xeIVDrK1HByZsNIXy/SSK8U26S2PVZ
2o14fWvKbJ1Aga8N6DuWY/D8P2mi3RABiuZgfkzmkL5idH/wSKfnFKdTgJzssdCc
1jZEGV5k5rFYcWOrJARHeP/tsnb/UxKBEsNtO7e3N2e/rLVnEyKVI0066hz7xZK/V
NBSpx3k3qj4XPK41IH7y2iQeIBBABAgAMBQJ0zqO8BQMAEnUAAAOJEJcQuJvKV618
2twH/0IzjXLxN45nviFiejc75a+i9ZSLlqR81sHL4GpEScFKI0a01T4IVAIIY2RKG+
MAs2eHmOUfKuwGs5jluRZ9RqKrc61sY0XQV9/7znY9Db16ghX04JjknOKs/fPi87
rvKkB/QxJWS8qbb/erRmW+cPNjBRxTFPS5JIWFWHA16ieFEpvdAgKV6nfVJVTqlr
jPdcnIA9CJN2SmUfX9Qx3SRc6ITbamlhjFnY6sCh6AUhxLI2flmq1xH9PqEy42Um
68prRqTyJ7Ioxlg/UDDkeeUcAg7TlviTz7uXpS3Wrq4zZo4yOpaJfLDR3pI5g2Zk
SNGTMO6aySE4OABt8ilPclPm6AmJASIEEAECAAwFak7yPFYFAwASDQAACgkQlxC4
m8pXrXzXiAf9FrXe0l9cPM+tYOWMLhv5gXjI2VUBaLxpyRXm/kJcmxInKlGCD3y
D4/FLHNU3ZcCz/uklPabZXWI006ewq0LWsRtklmJjWiedH+hGyaTv95Vkl0jRIBd
8nBaJ6M98r1jMBHTFwVvjQFVf4FLRJQZqHlvjCckq2Dd9BWPjGXvr/gpKkmMJYnk
/ftfZrCChb35NI19WRPohj9u808OPcQkVvZBcPwFGV5cEBZmAC94J7JcD8+S8Ik8
iUJMQLZ3cMZOBozovh86hj7KTSEBHLX1832z89H1hLeuLbnXoGLv3zeUFSxkv
lh35LhLzLQIMDQRXLuZzxGHMBpLhPyGWRJ4kBiGQQAQIADAUCTwQJFwUDABJ1AAAK
CRCXELibyletFABvB/9Cy69cjoQLGywITs3Cpg//40jmdhSAVxilJivP6J5bubFH
DJLVTx541Dv5h4hTGB2QuueQ4q1VCpSGW+rHcdhPyvmZGRzLrxdQqGh1Dv0Bod2c
3PJVSYPsrRSWCZJkHJOTVRBdjK4mkZb5aFTza+Tor9kxzj4FcXVd4KAS+hhQHYHc
Ar8tt2eOLzqdEFTULeGiSoNn+PVzvzdfhndphK+8F2jFQ2UKuc0107k0Yn9xZVx0
Og6fE1gStzLv7C5amWLRd8+hx+MN0G8MgNg1pBoExsEMMLPBYSUHa61xpDMNMuib
rIyVncE9X8QOhImt8K0sNn/EdbuldJNGYbDLt704iQeIBBABAgAMBQJPFdTCBQMA
EnUAAAOJEJcQuJvKV6184owH+wZ/uLpezXnSxigeH1sig72QEXMrNd5DVHCJdig3
bo+K5YmmN710/m5z+63XKUEWpd6/knaJobgckThzWftNeK1SSFQGPmoYZ9PEZnSU
7L+/dSUPExbj842G5LYagrCyMgt1xRywWEmbi72TKS/JOK0jLiOdvVy+PHrZSU0D
TVQ7cJh1BmPsbz7zzjmcI51+7B7K7RHZHq45nDL0IabwDacj7BXvBK0Ajqz4QyJ
GQUjXC7q+88I+ptPvOX1E5nI/NbiCJOMI6d/bWN1KwYrC80fZuFaznfQFcPyUaDw
yRaun+K3Kejj2wXecq+yMmLUep01TKSueOL50HD6hHH07W+JASIEEAECAAwFak85
bQsFAwASDQAACgkQlxC4m8pXrXwKpQgAlkbUsTr7nkq+haOk0jKpaHWEbRMEGMrB
I3F7E+RD0V6/8y4Jtn04EYDc8GgZMBah+mOgeINq3y8jRMYV5jvtZxv2MWFYUcjm
kVBKeqhl/pGEjmdm6t3D1Pv3Z+EMTMRmAocI981iY/go8PVPg/+nrR6cFK2xxn0
R8TackJBFeSfkkORg1tdZjjYv1B5ZIEkpplep15ahJBBq7cpYhTdy6Yk0S0J8w
EdffLsANxrRuWLRhWzZU7p9bFzfb/7OHC21dJnB7wKv5VvtgE+jjQw9tOKaf5hc
SgRYuF6heu+B25gc5Uu881o409mZ7oxQ6hDcn7JHvzh0rhmSN+Kid4kBiGQQAQIA
DAUCT0qQRQUdABJ1AAAKCRCXELibyletFC9UB/4o2ggJYMOCLXepP0GU8UKOh3+/
zmlDN7Qc4kY2iCtF1p1KHQaTgt5F1gRcFaiXcVv7WzGz/FnmxonR1leLl+kfRlwy
PPnoI/AWPCy/NO4Cl5KnjsSmsdDUpObwZ4KYsdilZR7ViJu2swdAIGNXBuWrlRJR
7CK4TAKrTeonRgVsrVx8Vt//8/cYj73Clq8oY/KK0iHiQrSwo44uyhdiFIAssjyX
n6/2E+w0zgvPexNSNROHQ8pjbq+NTY6GwKIGsaej3UTRwQ7psvKXz8y7xdzmOAr
/khGvxB5gjxk02pimjeia8v66ah6rbnojJMAovNUS4EHdHnulv4rovC8Kf9iieEi
BBABAgAMBQJVPVdsabQMAEnUAAAOJEJcQuJvKV618vVEIALFXPBzCA01SnQarBLzy
YMVZzumPvSXKnUHAO+6kjApXPJ+qFRdUaSNshzXvKY9Zryblu4ol/fLUTt0ClisD
IxD6L4GXEm4VYyC141P03bVsJnGITLFWQGHM27EmjVotid8Ch7kPq2EXr3dMRgzj
pdz+6aHGSufOdLTPXufDvW83bEWGarVutJKw+wIrcuRqQ+ucWJgJGwC4zeHjZad
Jx1XUm1X+BbI73uiQussyjhhQVVNU7QEdrjyuscaZ/H38wjUwNby1xDPB4I8quC1
knQ0wSHr7gKpM+E9nhiS14poRqU18u78/sJ2MUPXnQA6533IC238/LP8JgqB+BiQ
BTSJASIEEAECAAwFak9ng3cFAwASDQAACgkQlxC4m8pXrXxQRAF/UZlkkpFJj1om
9hIRz7gS+17YvTaKzspo+TBcx3C7aqKJpir6TlMK9cb9HGTh02Xp1N3FtQL72Nv0
6CcJpBURbvSyb4i0hrm/YcbUC4Y3eajWhkRS3iVfGNFbc/rHthViz0r6Y51hXX16
avkDv5CIFYWaf3BiUK0FnHrZiy4FPacUXCwEjv3uf8MpxV5oEmo8Vs1h4TL3obyUz
qrImFrEMYE/121kE8R5KwCaF8eFy156HL3PP190JMqBXzhwsFowCPuwjEM5w6sW
Ll//zynwxtlJ9CRz9c2vK6aJ8DRu3OfBKN1iiecNEynksDnNXErn5xXKz3p5pYdq
e9BLzUQCDYkBiGQQAQIADAUCT3inRgUDABJ1AAAKCRCXELibyletFGMKCADJ97qk
geBntQ+tZtKSFyXznAugYQmbzJld8U6eGSQnQkM40Vd62UZLdA8Mj1WKS8y4A4L2
0cI14zs5tKG9Q72BxQ0w5xkx1LASw1/8WeYebw7ZA+sPG//q9v3kIkru3sv64mMA
enZtxsykexRyCcumXljlAcLldrWJGUYE2Kl6uzQS7jb+3PNBl0Qvz6nb3YRZ+Cg
Ly9D41SIK+fpnV8r4iqhu7r4LmAQ7Q1DF9aoGaYvn2+xLgyWHxJAUet4xkMNOLp6
k9RF1nbNe4I/sqeCB25CZhCTEvHdjSGTD2yJR5jfoWkw09w8DZG1Q9WrWqki4hSB
10cmcv034pC1SJYziQeIBBABAgAMBQJPinQFBQMAEnUAAAOJEJcQuJvKV618CFEI

Signature Checking Using GnuPG

AJp5BbcV7+JBMRSvkoUcAWDOJSP2ug9zGw5FB8J90PDefKWCKs5Tjayf2TvM5ntq
5DE9SGaXbloIwa74FoZlqqlhMz4AtY9Br+oyPJ5S844wpAmWMFc6NnEPFaHQkQ+b
dJYpRVNd9lzagJP261P3S+S9T2UeHVD0JBgWIq9Mbs4lnZzWsnZfQ4Lsz0aPqe48
tkU8hw+nflby994q1wN0lk/u+I/1JbNz5zDY91oscXTR12jV1qBgKjYwCXxyB3j9
fyVpRl+7QnqbTWCICVFL+uuYpP0HjdoKNqhzEguAUQQL0B9msPTXfa2hg+32ZYg
5pzI5V7GCHq0K06u5Ctj3TGUJASIEEAECAAwFAk+cQEFAwASdQAACgkQ1xC4m8pX
rXzi7AgAx8wJzNdd7U1gdKmrAK//YqH7arSssb33Xf45sVHDpUVA454DXeBrZpi+
zEuo03o5BhAuf38cwfbkV6jN1mC2N0FzfpY4v7RxHKLyr7tr6r+DRn1LlgiX5ybx
CgY0fLAXkwcWUKGKABWkz9b/beEXaO2rMt+7DBUdpAOP5FNRQ8WLRWBcMGQiaT
S4YcNDAiNkrSP8CMLQP+04hQjJahxwCgBnksylciqz3Y5/MreybNnTOrdjVdsF0Oe
t0uL0iWXUZV1FfaGIdb/oBQLg+e1B74p5+q3aF8YI97qAZpPalqiQzWIDX8LX9QX
EFyZ3mvqzGrxkF0ocX1eNpGWT8fRuokBIgQQAQIADAUCT64N/QUADABJ1AAAKCRCX
ELibyletfdOGCACKfcjQ1SxrWlEUrYYZp0BP7DE+YdlIGumt516vBmxmt/50Ehqr
+dWuoioyC5tm9CvJbuZup8anWfFzTTJmPRPsmE4z7Ek+3CNMVM2wLynsLOt1pRFK
4/5RNjRLbwI6EtoCQfpLcZj//SB56sK4DoFKH280k4cplESPnoMqA3QafdSEA/FL
qvZV/iPgtTz7vjqkMgRxAIUM4fvKe3iXkAExGXtmgdXHVfoKmHrxJ2DTSvM7/19z
jGJeu2MhIKHyqEmCk6hLjxyCE5pAH59K1baQOP1bS28xlRskBAPm2wN+LOZwZc62
HhEreQ50inCGuubK0PqUQnyYc+1UFxrFpcliQEiBBABAGAMBQJPv91VBQMAEnUA
AAoJEJcQuJvKV618Agzh/iRFFC14qjvoqjilfi17yNPZVOMMO2H13Ks+AfCjrtHuV
aa30u50ND7TH+XQe6yerTapLh3aAm/sNP99aTxIuwrSlyKEoDs93+XVSGRqPbgBf
/vxv0yk0k3p6L9DxF0/w5cL8JrBhMZoJrEkIBfKwN8tWlcXPRFQvcdBYv3M3DTZU
qY+UHN0xHvSzsl+LJ0S9Xcd9C5bvYfabmYJvG5eRS3pj1L/y3a6y6hvY+JtnQAk
t05TdeHMigQH/zb8V9wXmDzME0un8LyoC2Jx5TpikQsJSejwk6b3coxVBIngku6+C
qDAimObZLW6H9xyYK10F0Js7j5bQZEWU070LBGjcmoQJASIEEAECAAwFAk/Rpc8F
AwASdQAACgkQ1xC4m8pXrXw49Qf/TdNbnun2htQ+cRwarszOx8BLEiW/x6PVyUQpZ
nV/0qvhKz1JUjM9hQPcA0AsOjhtCN6Cy8KXbK/TvPm9D/Nk6HWwD1PomzrJVfK2
ywGFiuTr+1luKSp7mzm5ym0wJs5cPq731Im31RUQU8ndjLrQ9YOf5FVL8NqmcOAU
4E8d68BmVCQC5MMr0901FKwKznShfpy7VYN25/BASj8dhynBYQErqToOJB6Cnd
JhdTlBfR4SirqAYZzG3XeqGhByytEHELx7FMWWFYhdntsnAvHYBbwqAzBs81f9Jd
Mhaf0VQU/4z10gVrRtXLR/ixrCi+P4cm/foQkqd6pwqWkaXt6okBIgQQAQIADAUC
T+NxIAUDABJ1AAAKCRXELibyletffBBCAC6+0TUJdCNaqOx0G1KViY6KYg9NCL8
pwNK+RKNK/NlV+WGJQH7qDMwRoOn3yogrHax4xIeOWiILrvHK006drS1DjsymIhR
Sm2XbE/8pYmEbuJ9vhH3b/FTChmSAO7dJskdWD3dvaY81SsuDDqPdTX8FzOfRXC
M22C/Yp70UG2A5svE1b+ylsmP4KmnVNAepEuPZcneMPFGop3haHg9Xv2+mj/btDB
Yr6p9kAgIY17nigtNTNjtI0dMLu43aIzedCYHQ1NhiB049jkJs54fMGBjF9qPtc
m0k44xykd1/JXWMDnUmtwKsChAXJS3YOciMgIx6tqYUtnDrP4I6qlrfriQEiBBAB
AgAMBQJP9T1VBQMAEnUAAAJEJcQuJvKV618J9wIAI11Id9SMbEHF6PKXRe1541E
pap5imMU/1GTj+9ZcXmlf8o2PoMMmb3/E1k+EZUaeSBOmjs8C2gwd5XFWRrlwAD
R1K/pg5XsL4h5wmN2F1lororrJXvqH427PLRQK9yzdwG4+9HTB0xjoS8qZT9pLyK
AJZzAydAMqyseRHgNo0vMwlgR4s4ojo+GcFGQHRf3IaUjvVfUP0mIj7afopFdIZmI
GaSF0TXBzqcZlchFv/eTbcLuIKRvLaDee5FgV7+nLH2nKOARCLvV/+8uDi2zbr83
Ip5x2tD3XuUZ0ZwXD0AQWcrLdmGblkxbGxvCtsaJHaLXWQ2m760RjiUcwVMEBKJ
ASIEEAECAAwFALAGYsFAwASdQAACgkQ1xC4m8pXrXwYVAgAvuvE16yGkniWolV
uHEusUv/+2GCBg6qv+IEpVtbtCCgiFjYR5GasSp1gpZ5r4Boc01bGdjdJGHTpyK8
xDli+6qZUWYhNRG2POXUVzCNE12hhouwPLOifcmTwAKU76TEv3L5STviL3hWgUR2
yEUZ3Ut0IGVV6uPER9jpr3qd603PeuFkwf+NaGTye4jioLay3aYwtZCUXzvYmNLP
90K4y+5yauZteLmNeq26miKC/NQu4snNFC1PbGRjHDLex9KDiAmTtOgN4WEq7srT
rYgtT531WY4deHpnGoPlHPuAfC0H+S6YwUmBgfcb6dV+Rrd8Ij6z3B/PcjmSUYf
OpdPtIkBIgQQAQIADAUCUBgtfQUADABJ1AAAKCRXELibyletFAm3CACQ1w21LfeG
d8RmITtsfnFG/sfM3MvZcJvfeAtsY3fTK9NiyU0B3yX0PU3ei37qEW+50BzqiStf
5VhNvLfbZR+yPou7o2MAP31mq3Uc6grpTV64BRikCmRWg40WMjNI1hv7AN/0atgj
ATYQXgnEw7mfFb0XZtMTD6cmrZ/A9nTPVgZDxzopOMgCCC1ZK4Vppq9FKdCYUaHpX
3sqndf+gpVIHkTCMgWLYQoeX5Nl+fgnq6JppaQ3ySZRUdr+uFUs0uvDRvI/cn+ur
ri92wdDnczjFumKvz/cLJA95TG2Jv1Jx3wecALsVqQ3gL7f7vr10MaqhI5FEBqdN
29L9cZe/ZmkriQEiBBIBCgAMBQJVoNxyBYMHhh+AAAJEEOz7NUmYpXLD1EH/2eh
7a4+8A11PLy2L9xcNt2bifLFFP2pEjcg6ulBoMKpHvutCgtX6ZPdHPm7uUOje/F1
CCN0IPB533U1NiOWIKndwNUJjughtORM+caMudYyc4kQm29Se6hMPdfyswXE5Bwe
PmoOm4xWPVOH/cvN04zyLuxdlQZNQF/nJg6PMsz4w5z+K6NGGm24NEPcc72iv+6R
Uc/ry/7v5cVu4h05+r104mmNV5yLecQF13cHy2JlmgIXHXPslTzbeJX7qqxE7TQh
5nviSPgdk89oB5jFxs4g1efXiwtL1P71bDlxHduomyQuH9yqmPZMbKJt9uZdc8Zz
MYsDDwlc7BIe5bGKfjJqAhwEEAECAAYFAlSanFIACgkQdzHqU52lCqLdvg//cAEP
qdN5VTKWEODfjDS4I6t8+0KzddWDacVFwKJ8RAo1M2Sk1DxnIvnyZsZd2VHp5Pq7
i4LYCzo51DkertQ6LwaQxc4X6myKY4LTA652ObFqsSfgh9kw+aJBBAyeahPQ8CDD
+Y123+MY5wTsJ4qt7Kffnzy78vLbYnVnrQ3/CboVix0SRzg0I30i7n3B01ihvY
5goy9ikjzZevejMEfjferCgoryy9j5RvHH9PF3fJVtUtHCS4f+kxLmbQJ1XqNDVD
hlFzjz8oUzz/8YXy3im5MY7Zuq4P4wWiI7rkIFMjTYSpz/evxkVlkr74qOngT2pY
VHLyJkqwh56i0aXcJmZiuu2cymUt2LB9IaMyWBNJjXr2doRGMAfjuR5ZaitmML

Signature Checking Using GnuPG

yZwix9mWVk7tkwIxmT/IW6Np0qMhDZcWYqPRpf7+MqY3ZYMk4552b8aDMjhXrnO
OwLsz+UI4bZa1r9dguIWI2C2b5C1RQ9AsQBpwg7h5P+HhRuFAuDKK+vgV8FRuzR
JeKkFqwB4y0Nv7BzKbFKmP+V+/krRv+/Dyz9Bz/jyAQgw02ultPupH9BGhlRyluN
yCJFTSNj7G+OLU0/14XNph500C7sy+AMZcsL/gst/TXciZrCcUApNTPDaenACpbv
g8OoIzmNWhh4LXbAUHCKmY//hEw9PvTZAlxKHgyJAhwEEgECAAYFAlJYsKQACgkQ
oirk60MpxUV2XQ//b2/uvThkkbeOegusDC4AZfjnL/V3mgk4iYy4AC9hum0R9oN1
XDR51P1TEw9mC1btHj+7m7Iq1a5ke5wIC7ENZiilr0yPqewGL5+LC98dz/L85hqA
wIoEOfMhrlaVbAZEj4yQTAJDA35vZHVsqmp87il0m+fZX04OBLXBzw86EoAAZ7Q
EoH4qFcT9k1T363tvNnIm3mEvkQ5WjE1R9uchJalq7hdlnQ1VkjFmPZrJK9f14z5
6Dt089Po4Sge48jDH0pias4HATYHsxW819nz5jZzGcxLnFRRR5iITVzi9qzshP7N
bUh3qxuWCHS9xziXpOcSZY848xXw63Y5jDjfpzupzu/KHj6CzXYJUEeqp9MluoGb
/BCCEPzdZ0ovyxFutM/BRcc6DvE6sTDF/UES21R0qfuwtJ6qJYWX+1BIgyCjv4o
RdbzxUleePuzqCzmwrIXtoOKW0Rlj4SceF9yCwUMBTGW5/nCLmN4dwlKw2RP2Eg
4ERbuUy7QnwRP5UCL+0ISZJyYUISfg8fmPIDqsetUK9Cj+Q5jpb2GXwELXWnIK6h
K/6jXp+EGEXSqdIE53vAFe7LwfHiP/D5M71D2h62sdIomUm3lm7xMOnM5tKlBiV+
4jJSUmrjCT62zo710+6iLgqmUUYLE1l6Ppvo8yuanXkYRCFJpSSP7VP0bBqIZgQT
EQIAJgUCTnc9dgIbIwUJEPzpwYLCQgHAWIEFQIIAWQWAgMBAh4BAheAAoJEIxx
jtTQcuHlU4AoIkjhdF70899d+7JFq3LD7zeeyIOAJ9Z+YyE1HZSznYi73brSciI
bIV6sbQ7TXlTUUwUGFja2FnZSBzaWduaW5nIGtleSAod3d3Lm15c3FsLmNvbSkG
PGJlaWxkQG15c3FsLmNvbT6IbwQwEQIALwUCTnc9rSgdIGJlaWxkQG15c3FsLmNv
bSB3aWxsIHNOb3Agd29ya2luZyBzb29uAAoJEIxxjtTQcuHlT0An3EMrSjEkUv2
9OX05JkLiVfQr0DPAJwKtLlycnLPv15pGMvSazv8JyWN3Ih1BBMRAGAdBQJHrJS0
BQkNMfioBQsHCgMEAxUDAgMWAgECF4AAEGkQjHGN01By4fUHZUdQRwABAa6SAJ9/
PgZQSPNeQ6LvV29yCALEBJOB7QCffgs+vWP18Jutdzc7XiawgAN9vmmITAQTEQIA
DAUCPj6j0QWDCWYAuWAKCRBJUOEqsNKR8iThAJ9ZsR4o37dNGyl77nEqP6RALJqa
YgCeNTPTEVY+VXHR/yjfy0bVurRxT2ITAQTEQIADAUCPkKAWWDCWIIiQAKCRC2
9c1Nxr0kP5aRAKCIaaegaMyiPKenmm8xeTJSR+fKQCgrv0TqHyvCRINmi6LPucx
Gkwfy7KIRgQQEQIABGUCP6zjrwAKCRCvxSNIEIN0D/aWAKDbUEgwwAFNH2n8gGJ
Sw/8lAuISgCdHMzLAS26NDP8T2iejsfUOR5sNriIRgQQEQIABGUCP7RDdwAKCRCF
lq+rMHNOZsbDAJ0WoPV+tWILtZG3wYqg5LuHM03faQcKuVvCmdPtro06xDzeeTX
VrZl4+GIRgQQEQIABGUCQ1uz6gAKCRCL2C5vMLLXh90AJ0QsqhdAgTak3SBnO2w
zuS0wiDIUwCdFExsdTxf1cL3Q4ilo+OTdrTW2CIRgQQEQIABGUCRPEzJgAKCRD2
ScT0YJNTDapxAKCJtqT9LCHFyfwKNGGBgKjka0zi9wCcCG3MvnbZDUQdVebudUZ
61Sont+ITAQEQIADAUCQYHLAQWDBiLzIwAKCRAYWdAfZ3uh7EKNAJwPywk0Nz+Z
Lybw4YNQ7H1UxZycaQcPvH4P5CHGjeYj9SX2gQCE2SNx+ITAQEQIADAUCQYHL
NAWDBiLZWAACRCBwvfr4h02kiIJAJ0VU1VQHzF7yYVeg+bh3lmg900kwCeJI8D
9mx8neg4wspqvgXRA8+t2saITAQEQIADAUCQYHLyGwDBiLZKGAkCRBrcOzZXcP0
cwmqAJsfJ0vY9c5eA/zyMrOz1uPB6pd4QCdGyzgbYb/eoPu6FMvVI9PVIENzReI
TAQEQIADAUCQDEJAuWDBQraAAKCRB9JcoKwSmmwmJVAKCG9a+Q+qjCzDzDtZKX
5NzDW1+W+QCeL68seX80oiXLQuRlifmPMrV2m9+ITAQEQIADAUCQitbugWDBXLI
0gAKCRDmG6SfJEU5q/MTAKCTmvlCQtLkLzD0sYdWVLHXJrRUVgCfmdes6AdpWIn
U0/yvYjglxlyiuqITAQSEQIADAUCQCPzOgWDB3pLUGAKCRA8oR801Pr4YSZcAJwP
4DncDk4YzvDvnRbXW6SriJnlyQCdEy+d0CqfdhM7HGUs+PZQ9mJKBKqITAQSEQIA
DAUCQD36ugWDB2ap0gAKCRDyl1xj45xlnLlFAK0NzCVqrBTDWR25cUss14RRoU
PACeLpEc3zSahJUB0NNGTnlpw1TczlCITAQSEQIADAUCQ4KhAWDBpaaCAAKCRA5
yiv0PWqKX/zdAJ4hNn3AijtcAyMLrLhLzQvib551mwCgw6FEhGLjZ+as0W6811uc
wZ6PzW+ITAQSEQIADAUCQc1NAWDBSP/WAAKCRAdEcCFfIOFqOMkAJwPUDhS1eTz
gnXclDKgf353LbjvXgCeLCWyyj/2d0gIk6SqaP12UcWrqiITAQTEQIADAUCPk1N
hAWDCVdCAAKCRATu3a/rdTJMwUMAKCVpkbklUp/kyPr1sVKU/Nv3b0TZACfW5za
HX38jDCuxsjIr/084n4kw/uITAQTEQIADAUCQdeAdgWDBc0kFgAKCRBm79vIzYL9
Pj+8AJ9d7rvGJChZTCSYVnaStv6jP+AEACeNHa5yltqierBCCcLcAcGqYK81omI
TAQTEQIADAUCQhIBDgWDBYwJfGAKCRB2wQmcojFuoADuAJ9CLYdysef7IsW42UfW
hI6HjxkzSgCfeEpXS4hEmmGicdpRiJQ/W21aB0GIZQQTEQIAHQULBwODBAMAWID
FgIBAheABQJLcC/KBQkQ8/OnABIHZUdQRwABAQkQjHGN01By4fWw2wCeJilgEarL
8eEYfDdYTyrdqE45HkoAnjFSZY8Zg/iXeErHI0r04BRukNVgiHsEMBEcADsFAkJ3
NfU0HQBPb3BzLi4uIHNOB3VsZCBoYXZlIGJlZW4gbG9jYWhIEknbsAqc28qIHNO
dXBpZC4uLgAKCRA5yiv0PWqKX+9HAJ0WjTx/rqgouK4QCROV/2IOU+jMQQCfYSC8
JgsIIE8aiyuStTdYrk0VWCijwQEQIATwUCRW8Av0gdAFNOB3VsZCBoYXZlIGJl
ZW4gYSBsB2NhbCzbaWduYXR1cmUsIG9yIHNVbWV0aGluZyAtIFdURiB3YXMGSSB0
aGlu2luZz8ACgkQOcor9D1qil+g+wCfcFWoo5qU14XTE9K8th3Q+xGweYYAnjii
KxjtOx0c0ls+BlqXxbFZ9uqBsiQIiBBABAgAMBQJBGcuFbYMGIttkHAAoJEKj5s5m
oURoqC8QAIIISudocbJrhrTAR0OPomsReyp46Jdp3iLlLoFDGcPfkzSBWWh8L+cJjh
dycIwwSeZlD2h9S5Tc4EnoE0khsS6wBpuAuih5s//coRqIiILKEdhTmNquLkCH5m
imCzc5zXZDWBhPzLr2InGsZmuh2QCwAkB4RTBM+r18cUXMLV4YHYkjIVaDhSiPP/
MKUj6rJNSUDmDq1GiJd0jySjtCFjYADlQYSD7zcd1vpqQLThnZBESvEoCqumEfOP
xemNU6xAB0CL+pUpB40pE6Un6Krr5h6yZxYZ/N5vzt0Y3B5UUMkgYDspjbulNvaU
TFiOxUe3gJvXc1+h0BsXm7FwBZnuMA8LEA+UdQb76YcyuFBCROhmcEUTiducLu84

Signature Checking Using GnuPG

E2BZ2NSBdymRQKSinhvXsEWLH6TxmlgtJLynYsvPi4B4JxKbb+awnFPusL8W+gfz
jbygeKdyqzYgk j3M79R3geaY7Q75Kx1lUogioKcbI5VZvg470QCWEEERnejqEAdx
EQiWGA/ARhVOP/110LQA7Jg2PlxTtrBqqC2ufDB+v+ jhXaCXxstKSW1lTbv/b0d6
454UaOUV7Risn39pE2zFvJvY7bwfiwbUJVMYLM4rWJAE0JLIDtDRt2h8JahDObm
3CWkpadjw57S5v1c/mn+xv9yTgVx5YUfC/788L1HNKXfeVDq8zbAiQIiBBMBAGAM
BQJCNwoCByMFBZpWAAoJENjCCglaJFFPIT4P/25zvPp8ixqV85igs3rRqMBtBsJ+
5EoEW6DjnlGhoi26yflnasC2frVasWG7i4JImOU3WfLZERGDJR/nq10CEqsP5gS3
43N7r4UpDkBsYh0WxH/ZtST51lFK3zd7XgtxvqKL981/OSgi jH2W2SJ9DGpjtO+T
iegq7igtJzw7Vax9z/LQH2xhRQKZR9yernwMSYAJ72i9SyWbK3k0+e95fGnlR5pF
zlGq320rYHGd7v9yoQ2t1klsAxk6e3b7z+RiJG6cAU8o8F0kGxjWzF4v8D1op7S+
IoRdB0Bap0lko0KLyT3+g4/33/2UxsW50BtfcqvYNjvU4bZns1YSqAgDOOanBhg8
Ip5XPLDxH6J/3997n5JNj/nk5oJfd8nyfe/5TjflWNiput6tZ7frEki1w16pTNbv
V9C1eLUJMSXfDZyHtUXmiP9DKNpsucCUEBKWRKLqnsHLkLYydsIeUJ8+ciKc+EW
HFXEY+Ml72cXAaz5BuW9L8KHnzZZfez/ZJabiARQpFfjOwAmhZJ9r++TEKRLER96
taUI9/8nVPvT6LnBpcM38T6dJ639YvuH3ilAqmPPw50Yvg1IEe4BUYD5r52Seqc
8XQowouG0uBX4vs7zWfYuA/s9ebfGaIw+uJd/56Xl9116q5CghqB/yt1EceFEnF
CAjQc2SeRo6qzx22iEYEEBECAAYFAkSAbycACgkQCwYeUx5vWDCACfQsVk/XGi
ITfYFVQ3IR/3Wt7zqBMAOnhso/cX8VUfs2BzxPvvGS3y+5Q9iEYEEBECAAYFAkUw
ntcACgkQOI416LNBlYkyFgCbCw5gIii0RTDJsdNiuJdCu/NPqEAniSq9iTaLjgF
HZbaizUU8arsVCB5iEYEEBECAAYFAkWho2sACgkQu9u2hBuwKr6bjwCfa7ZK60+X
mT08Ssysg4DEoZnK4L9UAoLWgHuYg35wbZYx+ZUTh98diGU/miF0EEExECAB0FAj4+
owwFCQlMAyAFcWcKAWQDFQMCAXYCAQIXgAAKCRcMcY07UHLh9XGOAJ4pVME15/DG
rUDohtGv2z8a7yv4AgCeKIp0jWUWE525QocBwms7ezxd6syIXQQTQIAHQUCR6yU
zUWJDTBYqAULBwDBAMVAwIDFgIBAheAAoJEXxjTtQcuH1dCoAoL6Rtsd9K3N
7NOxcp3PYozH2oqzAKCFHn0jSqxk7E8by3sh+Ay8yVv0BYhdBBMRAGAdBQsHCgME
AxUDAGMwAgECF4AFakequSEFCQ0uFRUACgkQJHGNO1By4fUdtwCfRncueXikBMy7
tE2BbfwEYTLBTFAAnifQGbkmcARVS7nqauGheLED/vdgiF0EEExECAB0FCwCkAWQD
FQMCAXYCAQIXgAUCS3AuZQUJEPpyWQAKCRcMcY07UHLh9aA+AKCHDKOBKBrG8tO
g9Blub3LfhMvHQCeIO0otlhhHULsTIXAUrD8+ubIeZaJARwEEgECAAYFAkVCIgMA
CgkQ3PTrHsNvdI8eQgf/dSx0R9K1ozz8iK79w00N0sdoJY0Na0NTFmTbqHg30XJo
G62cXYgc3+Tjnd+pYhYi5gyBixF/L8k/kPVPzX9W0YfWChZDsftw0iDvmGxOswiN
jzSo0lhWq86/nEL30Kh19AhCC1XFNRw8WZYq9Z1qUXHHJ2rDARaedvpKHOjzRY0N
dx6R2zNyHDx2mlfCQ9wDchWEuJdAv0uHrQ0HV9+qx71W/Q3L/V5AuU0tiowyAbBL
PPYrB6x9vt2ZcXS7F0Wt8SfQ1i8W2QDQ/Toork4YwBiv6WCW/ociy7paAoPOWV/NF
2S6hDispeebk7wqpbUj5k1Dmwr1gB/jmoAXWENbsYkBIgQQAQIADAUCSSpooAUD
ABJ1AAAKCRCELlibyletffOMCACp+OVZ71H/cNY+373c4FnSI0/S5PXS0ABgdd4
BFRWFWRkWBExBGc8szfH0zVewkzV96iyHbpddeAOAkEA40VPW1MMFCmlHxi2s9/N
JrSrTPVfQOH5FR9hn7Hbpq/ETw0IoX1FKo7vndMnHZNfEnI+PDXLcdMYQgljYzhT
xER4vYUUKu8ekSshly4zOX7XSJxwqPUPvps8qs/TvojIF+vDJVgFYHvkgs+shp8
Oh/exg9vKETBlgU87Jgsqn/SN2LrR/Jhl0aLd0G0iQ+/wHmVydQUMFaCZwk/BKNa
XPzmGZEUZ3RNBYa19Mo7hcE3js76nh5YmxFvxbTggVu4kdFkiQEiBBABAGAMBQJK
M06IBQMAEnUAAAOJEJcQuJvKV618F4gH/innejIHffGMk8jYix4ZZT7pW6ApyoI+
N9Iy85H4L+8rVQrtcThyq0VkcN3wPSwtfZszUF/0qP6P8sLJNJ1BtrHxLORYjJPm
gveeyHPzA2oJ16imqWUTIw822fyjY/azwhvZFzxmVbFJ+r5N/Z57+Ia4t9LTSqTN
HzMUYaXKDaAqzZeK7P0E6XUaaeygbjWjBLQ100ezozAy+Kk/gXApMDCGFuHSFe7Z
mgtFcbXLM2XFQPMUooETD2R8MUsd+xnQsff/k6pQOLxi+jUESWSr/iqmv1k6gZ4D
pemBjuhcXYlxjYjUaX9Zmn5s+ofF4GFxRqXoY719Z+tCM9AX371m6S+JASIEEAEC
AAwFAkpEcgoFAwASdQAACgkQ1xc4m8pXrXz2mgf/RQkpmMM+5r8znx2TpRAGHi5w
ktvdFxlVPaOBWE28NDWTrpcMqo9kzAiuvEQjVNiHbP21wR3kvnQ84rTAH0mlC2I
uyybgppqwzOUL+Wi0o+vk8ZA0A0dStWRN8uqneCsd1XnqDelrvqC4/9yY223tLmA
kPvz54ka2vX9GdJ3kxMwewhrVQSLCktQpygU0dujGTDqJtnk0WcBhVF9T871v3W2
eGdPieLzHU5trXezmGFj21d56G5ZFK8co7RrTt4qdZnt80glh1BTGmhlzjMPLTe
dcMusm3D1QB9ITogcG94ghSf9tEKmmRJ6OnnWM5Kn9KcL63E5oj2/1Y9H54wSYk
IggQAQIADAUCS1Y+RwUDABJ1AAAKCRCELlibyletfoOQB/odyJBiBjgf+8d3yNID
pDktLhZYw8crIjPBvdOgX12xaUYBTGcQITRVHSGgzfDA5BQXeuUwhpL4QB0uz1c
EPPwSMiWiXlBtWf5q6RVf3PZGJ9fmFuTkPRO7SruZeVDo9WP8HjBqT0LukYf566e
grzAYR9p74UgWftpDtmrqrRTobiuvsFBxosberCvEQCrN0n+p5D9hCVB88tUPHnO
WA4mlduAFZDxQWTApKQ92frHiBqy+M1JFezz2OM3fYN+Dqo/Cb7ZwOAA/2dbws7o
y4sXEhbWonjsgkPQwFYB23tsFUuM4uZwVEbJg+bveglDSDStbd1fgArXSL/0+ak
lFchiQEiBBABAGAMBQJKaAQEBQMAEnUAAAOJEJcQuJvKV618rH0H/iCciD4U6YZN
JBj0GN7/Xt851t9FWocmcaC+qtuXnkFhplXkxZVOCU4VBMS4GB0qfIvagbBTyFV4
Di+W8Uxr+/1jiu3l/HvoFxdwNkGG6zNBhWSjdwQpGwPvh5ryV10fLX/mgQgdDmx
vqz5+kFDUj4m7uLaeuU2j1T01R4zU0yAsbt7J3hwfjqCXHoc9bm5nvJwMrSm+sdC
TP5hJUlwHr9mTe8xuZv76sO/w0P4AqIMxjC9W7pT9q0ofG2KStwt7wFbh05sbG4U
QYOJe4+Soh3+KjAa1c0cvmIh4cKX9qfCWwhhdeNfh1A9VTHhnl5zTv/UjvntQjh1
H/FqleBSKcSjASIEEAECaAwFAkp5LgoFAwASdQAACgkQ1xc4m8pXrXwY6wgAg3f8
76L3qDZTYLFAW53pXB18GsUr1DEkTLEDZMKDM3wPmhaWBR1hMA3y6p3aaCUyJII

Signature Checking Using GnuPG

BEneXzgyU9uqCxCXpC78d5qc3xs/Jd/SswzNYuvuzLYOW5wN5L31SLmQTQ8KqE0uo
RynBmtDCQ4M2UKifSnnv+0+3mPh85LVAS481GNpL+VVFcyTKesWnu40+98Yg6L9NG
WwRTfsQbcDokZo44Jz7Y7f81ObC4r/XlDgPj2+d4AU/plzDcdrbINOyprs+7340e
cnaG04Lsgd19b1CvcgJgltrQuu3kRvd+Ero2RYpDv6GVK8Ea0Lto4+b/Ae8cLXAh
QnaWQCEWmw+AU4Jbz4kBIgQQAQIADAUCSo5fvQUDABJ1AAAKCRCXELibyletfa08
B/9w8yJdc8K+k07U30wR/RUG3Yb2LBDygy09lmVsyB0RGixBDXEP0XBqGKAXiV1
QSMAXM2VKRsuKahY2HFkPbyhZtjbdTa7Pr/bSnPvRhAh9GNWvVrg2Kp3qXDd jv9x
ywEghKVxcEIVXtNRvpbgRokmHzIExvUQck5DM1VwfrEeYIoXgs4035WADhVMdngQ
S2Gt8P2Wau/p8EZHFgG6X8Kt0LD68zGboaJe0hj2VDC+Jc+KdjrFE3fW5IToid/o
DkUaIW6tB3WkXb0g6D/2hrEJbX3headChHKS8eQdOR9bcCJDhhU8csd501qmrhC
ctmvlpeWQZdiQdk6sABPWeeCiQEiBBABAgAMBQJKoBJHBQMAEnUAAAoJEJcQuJvK
V618M18H/1D88/g/p9fSVor4Wu5WlMbg8zEAik3BIxQruEFWda6nART6M9E7e+P1
++UHZsWys619R0PwXRLG1Yy9jLec2Y3nUtB20m65p+IVEKR2a9PHW35WZDV9dOYP
GZabKk01clLeWLVgp9LRjz+AeRG+1jHqsULXr0ldwewLTB/gg9I2vgNv6dKxyKak
nM/GrqZLATAq2KoaE/u/6lzRFZiZznLtjzh8X7+nS+V8v9IiY4ntrpkrbvFk30U6
WJp79oBIWwnW/84RbxutRoEwSar/TLwVRkcZyRxeJTapbnLgnQ/1D01o1d7+Vbjd
q/Sg/cKHHf7NthCwkQNScnHL0f5lqZCJASIEEAECAAwFAkqoEAAFAwASdQAACgkQ
lxC4m8pXrXwE/Af/XD4R/A5R6Ir/nCvKwCTKJmala jssuAcLEa2pMnFZY0/8rzLO
+Gp8p0qFH9C4LFWAQNvR5q6X/swuROf4zx1jSvNcdlQVaAfJZDEGJ5GXzsPpLrv
SAI9jS3LL7fSWDZgKuUe0a4qx7A0NgyGMUYGhp+QlRFa8vWEBI9fAnd/0mMqAeBV
qQy0H0X1FiWlCa2Jn4NKfuMy9GEvRddVlB1LvoNVtXPNzeeKMYNb9JdxlMFWssy
COBP2DayJKTmJvqPEc/YOjOowoN5sJ/jn4mVSTvvlTooLiReSs6GSCAJMVxN7eYS
/Oyq6IulJDCJvM8N2WixAZtAVgF80A7CWKKVykBIgQQAQIADAUCSrniHiQUDABJ1
AAAKCRCXELibyletFpChB/9uEct1ldZeNuFsd0/RuYRUUVlrrhJE6WCcOrLo9par
rPbawbKbMjSzb0MygJXGvcC06mPNuquJ7/WpxKsFmfg4vJBPlADFKtgrUY9BLzjC
eotWchPHFBVW9ftPbaQVlSUu7d89NLjDDM5xrh80puDIapxoQLDoLrh3TlkpZx56
jSwv0geLFUMbXazmKjJSyL4XdhlaqzgUbrEd7Xf2ICZuh0sV6V7c/AwWtjWEGESa
HZaiQDyWzwbC18GwrMLiAzGwB/AscFDQRcZKJDjL+Ql8YT6z+ZMvr8gb7CIU5PKY
dhi1f2UVTQWLaoW71NRCQQAQcGjK3IMIz7SO/yk4HmVUIQEiBBABAgAMBQJK3gJ
BQMAEnUAAAoJEJcQuJvKV618jkEH+wb0Zv9z7xQgpLMowVuBFQVu8/z7P5ASumyB
PU03+0JVXSHBh1CKQK7n1m1mfhuGt2fCxxShSU6LzXj36rsKRY531GZ9QhvfFUQH
3Xb2IQLIJC4UKjG2jSSCdcuA/x98bwp2v7003rn7ndCS16CwXnrV3geQoNipRKMS
DajKPPzV1RiZm8pMKqEb8WSw352xWoOcxuffjlsOewvJ85SEGCAZ9tmIlkZoc7Ai
QONDvii9b8AYhQ60R1QC0HP2ASSmK0V92VeFPxHmAygdDQgZNVtbVxgnnt7oTNEU
VRXNY+z40fBArp7R+cTsvijDRZY4kML1n22hUybwoxUEVjzV2+JASIEEAECAAwF
Akrv0lQFAwASdQAACgkQlxC4m8pXrXxrPAgArXiNgZiRnuBhfNcXlzkzCHLx5wnV
e4SmtPbwzTwW7+qk7d4l9h1WtdImISORINzo7f4ShSUzJX2GciNaXhaHR07+y50
Zbu82jQb09aQqj/nibKYuqxqUrobtEm+DuYz3JUQZm2PsPcHLS8mX9cxvrJUncPG
nXEV0DRaq71LSWDPrtkvBbp6i38aY3sIhYgz8wM5mlszKdtjywmBYcFehIdozt9z
hm7wZshzRWQX1+Rf/pIsnk+OzBia34crSemTnacv/B7278z2XAYziPNFuqz0xu+
iltOmYmayfNWAmumuw9NcuwWmlth6Mc2HLrpo0ZBheJ6iuDMPsHnwgdb/4kBIgQQ
AQIADAUCSwd2gUDABJ1AAAKCRCXELibyletFp6tB/4mlw0BtlkJgtS6E+B/ns14
z4A4PGors+n+MYm05qzvi+EnDF/sytCmVcKeimrtvDcfoDtKAFfVjJcYXfnJdGWM
Pu0SJMRL5KKClrAKwZmU/saxOgoB5QLNw+DHpteJ3w9GmWlGxIqG1r15WC5duzBC
y3FsnjJYG3jaLnH009yXXb5h0kUTORfUKdvAr1gxF2KoatZWqGoaPPnHoqb88rjt
zk8I7gDqoXnz8wLxa0ZYvfTC/McxdwTrwXLft+krmMQ18iIEZene2hvVLNVuluU
oiWLeHA8iNCQ4W4TDLc1mCncjGTMX/MN41uLH0C9Ka4R6wEaqj4LPdklB/1TV+Q
iQEiBBABAgAMBQJLEyGrBQMAEnUAAAoJEJcQuJvKV618naIH/2t9aH5mBTKBN6fU
qhrf79vIsjtI/QNS5qisBISZMX3/1/0Gu6WnxkPSfdCUJMWCjMcnvJ7KU2wxTHHG
VpAStd9r2afUNxRyqZwzwytktuZok0XngAEDYDDBS3ssu2R4uWLCsC2ysXEQ0/5
tI5YrTWJZrfeIphTaYp5hxrMujvqy3kEwKKbiMz91cDeiLS+YCBcalj5n/1dMYf7
8U8C6ieurxAg/L8h6x25VM4Ilx4MmG2T8QGtKkUXd+Fd/KYwfm0LE5LLPknf0Hhw
oVs1PXeinp4FsHK/5wzviv4Yzpzutqs9N1KcMsa4IuuPOB0FDf0pn+OFQbEg9QwY
2gCozK+JASIEEAECAAwFaksjTdqFAwASdQAACgkQlxC4m8pXrXwlogf/XBGbXRVX
LMaRN4Scz0jwT3/tUCriTkb3v+zKjRG90zFhYAccjn7w+7jKQicjq6quQG1EH2X4
/Su6ps1LDLqGHHhiJW3ZhxQScLZmhdAYsh2qG4GP/UW3QjXG7c61t+H3o1vWg2cr
wqCxxFZAgkAAkr9xcHWFZJEQeXoob6ccZOBAUnHSANdmC6s51UxXYa2bml7Q3UB4
4KcZDvAfBpZKJOW9k0qb3lc1lzx+vGdyZFbm4R0+3LPp/vT0b3G1Sbbf91U1GOXh
VaphrgFfa76dmjfhCkPpLXAK1VSIU/aPGAefduTFMdlSzdMj5AULjGcszBDLR
pLlPvxvQa0ZpgIkBIgQQAQIADAUCSycmkgUDABJ1AAAKCRCXELibyletFh1NCACp
lYespiHFQt2alcsE5zgfETEHHic8Ai6pNku9HT4TeWfHEDe5QqfYcpJLrQvBXS
kSvxEittbYrdv+e+j5Z+HyHjiG8naQBL6qy9eHqQE4+d7gYs6DTk7sG9ZMYphREb
ltzD+F4hVCQDLTL8Lnr0eVFN7ehqECSdCG8/Qyti+1/0M902/Yn+mz0ilOiuUdWJ
9x6LpaIIntblgsYDEylLjwGIzmIOr5Kh9wYoV4vnNzFbx01uriW0B7iaPjIESbt
OOKp7wx2aX+DM3N9F3BtaIY8XnzcNOM83SNsgmgrZljpQlUnNqIhNM8DupQ+I
Wov5gt16pTC7CgeVTyRiQEiBBABAgAMBQJLOGXuBQMAEnUAAAoJEJcQuJvKV618
1l4IAKJ9mm4jB0c8fe9+uDI8eCJRbzNbvXm8zWzpa8GUTQAakwxoKv332QP1Wa1P

```
odni/e3EMhsSRE0ZJJv79YqGxGRBTE9Kb/VjM34nas4XSnXKW28XWhKyIw+XwQAi
nY2swFHh+83Htr/mwTdTdfS2aEYL2zboBvd/JZCdhOGU2GH737S/3uEczoKkfVQ/w
OTM8XlxWwlyWqx23k/DsGcuDs9lA2g7Mx7DSqBtVjaTkn9h0zATzXLDkmP4SAUVj
cZ83WdpPre5WnizZjdXlBMM5OCexp5WpmzyHLTnaBFK4jEmnsk5C2Rnoyp8Ivz6g
EcgltrBExijRw++d2TFYlJwLktijASIEEAECAAwFAktKMicFAwAsdQAACgkQlxC4
m8pXrXxqHQgAuYY5scKrhOm/GS9EYnyC9494l0l06iytU0CpE6oBC3lM3hfX/Dbj
UbcS5szZNU+2CPY04uqJLZ7suN7+tTjG6pZFFMevajT9+jsL+NPMP8RLdLOVYmb1
TmSQGNO+XGEYaKYH5oZIEIW5AKCgi2ozkdFlBBLAx7Kqo/FyybhkURFECvEyVmGf
3KLV7IiIX/fYlfoCMCJ/Lcm9/1lSfBl8Nvng66Xd533DKoHjueD3jyaNAVl02mq/
sIAv++kntvOiB3GDK5pfwhZ78WWiCpsWzpe5gzAnzJlY0WEigRo0PVLu3cL00jLG
23d+H/CbfZ8rkaJhJECDF7YVmp0t0nYpYkBIgQQAQIADAUCS1v+ZgUDABJ1AAAK
CRCXELibyletFNS/CACqt2TkB86mjqM+cJ74+dWBvJ2aFuURuxzm95i9Q/W/hU08
2iMbC3+0k2oD8CrToE6lP+3oRyLjv/UEDUNzLncNe2YsA9JeV+4hvPwH5Vp30ml3
089fCKZUbqslXNKKHlWYU+zAaZXEUGRmRz0HbQIEAMOWF4oa226uo1e4wslJhc+
F3E/ApCRYFBqBUdL05hapQLditYpsBjIdiBGpjzidMLE2wX2W4ZpAdN0U6BIYIqR
mTPjbskVzS9kSWFmfHqgnBDKEYJpVZgELsN52rYClSDeGeiuKxlzjVov9MMhYMWa
Zo3R5o3F2iIM/BK6FbC252lf/Mhu3ICuXujNBZNYiQEIbBABAgAMBQJLbSH4BQMA
EnUAAAoJcJcQuJvKV618kd0IAJLLwDH6gvgAlBFklQJXqQxUdcSOOVMAwtlHgWoy
ozjgomZZBkRL8dtCDR9YBMcj5czcQ3qpmLjDppXhKB+kJV2lUXfDMSFXwJ4wLfiS
8FNnXw8H5U01oBkGH/Ku6ngL9Vwt+MjYHtCWkw9QueUKZnDudX9qIzLAI+mwSTu
A6+fY4VWig40AA0v3exaQM55YR/UhlKunpGG9o8Qkq77dMEbTmPomBoLbOMRB3Dd
MAvVU6G2l6Pcb7KobVCu0Bnb6batXARV/G8sw+nzjfJl6fr/KobZT2A6m+Jrjqk4dl
F14lJLz1605JGUPArYn2G2ddBdSAY7dtFSVhWWiWC9n88q5Ag0EPj6jHRAIAO/h
iXBWzHWOMLJ7T54x/axeDdqnlrBDF5cWmaCWHN2uJNNlGpx5emoU9v7QStsNUCOGB
bXke04Ar7YG+jtSR33zqNh3y5kQ0YkY3dQ0wh6nsl+wh4XIY/3TUZVtmdJeUBRH
JlFVNFYad2hXlguFI37NylPoZAFsx082g+XB/Se8r/+sbmVcONdcdTeFKrE3FjLt
IjNQCxC6l9Q20y8KDxG/zvUZG3+H5i3tdRMYGgmuD6gEV0GXOHYUOpzLeitl+AA0
bCk36Mwbu+BeOw/CJW3+b0mB27hOaf9aCA855IP6fJFvtxcblq8nHIqhu3Dc9tec
s19/S1xZ5S8y1G/xERsAAwUH/i8KqmvAhq0X7DgCcYputwh37cuZlHOa1Ep07JRM
BCDgkdQXkGrsj2Wz7Aw/TGdWwkmn2pxb8BRui5cfcZF07c6vryi6FpJuLucX975
+eVY50ndWkPXkJlHF4i+HJwRqE2zlin/RHMs4LJcWxQvvdJ43EE3A06eiVFbD+qA
AdxUfoOeLbLkNBHPG7DPG9xL+Ni5rke+TXShxsB7F0z7ZdJjZOG0JODmox7IstQT
GoaU9u4loyZTIxPiFidJoIZCh7fdurP8pn3X+R5HUNXMr7M+ba8lSNxce/F3kmH
0L7rsKqdh9d/aVxhJINJ+inVDnrXWVoXu9GBjT8Nco1iU9SIVAQYEQIADAUCTnc9
7QUJE/sBuAASB2VHUEcAAQEJEIxxjTtQcuH1FJSAmwWK9vmwRJ/y9gTnJ8PWF0BV
roUTAKClYAhZuX2nUNwH4v1EJQHdQYa5yQ==
=HfUN
-----END PGP PUBLIC KEY BLOCK-----
```

To import the build key into your personal public GPG keyring, use `gpg --import`. For example, if you have saved the key in a file named `mysql_pubkey.asc`, the import command looks like this:

```
shell> gpg --import mysql_pubkey.asc
gpg: key 5072E1F5: public key "MySQL Release Engineering
<mysql-build@oss.oracle.com>" imported
gpg: Total number processed: 1
gpg:          imported: 1
gpg: no ultimately trusted keys found
```

You can also download the key from the public keyserver using the public key id, `5072E1F5`:

```
shell> gpg --recv-keys 5072E1F5
gpg: requesting key 5072E1F5 from hkp server keys.gnupg.net
gpg: key 5072E1F5: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
1 new user ID
gpg: key 5072E1F5: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
53 new signatures
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:          new user IDs: 1
gpg:          new signatures: 53
```

If you want to import the key into your RPM configuration to validate RPM install packages, you should be able to import the key directly:

```
shell> rpm --import mysql_pubkey.asc
```

If you experience problems or require RPM specific information, see [Section 2.3.4, “Signature Checking Using RPM”](#).

After you have downloaded and imported the public build key, download your desired MySQL package and the corresponding signature, which also is available from the download page. The signature file has the same name as the distribution file with an `.asc` extension, as shown by the examples in the following table.

Table 2.1 MySQL Package and Signature Files for Source files

File Type	File Name
Distribution file	<code>mysql-standard-5.5.58-linux-i686.tar.gz</code>
Signature file	<code>mysql-standard-5.5.58-linux-i686.tar.gz.asc</code>

Make sure that both files are stored in the same directory and then run the following command to verify the signature for the distribution file:

```
shell> gpg --verify package_name.asc
```

If the downloaded package is valid, you will see a "Good signature" similar to:

```
shell> gpg --verify mysql-standard-5.5.58-linux-i686.tar.gz.asc
gpg: Signature made Tue 01 Feb 2011 02:38:30 AM CST using DSA key ID 5072E1F5
gpg: Good signature from "MySQL Release Engineering <mysql-build@oss.oracle.com>"
```

The `Good signature` message indicates that the file signature is valid, when compared to the signature listed on our site. But you might also see warnings, like so:

```
shell> gpg --verify mysql-standard-5.5.58-linux-i686.tar.gz.asc
gpg: Signature made Wed 23 Jan 2013 02:25:45 AM PST using DSA key ID 5072E1F5
gpg: checking the trustdb
gpg: no ultimately trusted keys found
gpg: Good signature from "MySQL Release Engineering <mysql-build@oss.oracle.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: A4A9 4068 76FC BD3C 4567 70C8 8C71 8D3B 5072 E1F5
```

That is normal, as they depend on your setup and configuration. Here are explanations for these warnings:

- *gpg: no ultimately trusted keys found*: This means that the specific key is not "ultimately trusted" by you or your web of trust, which is okay for the purposes of verifying file signatures.
- *WARNING: This key is not certified with a trusted signature! There is no indication that the signature belongs to the owner.*: This refers to your level of trust in your belief that you possess our real public key. This is a personal decision. Ideally, a MySQL developer would hand you the key in person, but more commonly, you downloaded it. Was the download tampered with? Probably not, but this decision is up to you. Setting up a web of trust is one method for trusting them.

See the GPG documentation for more information on how to work with public keys.

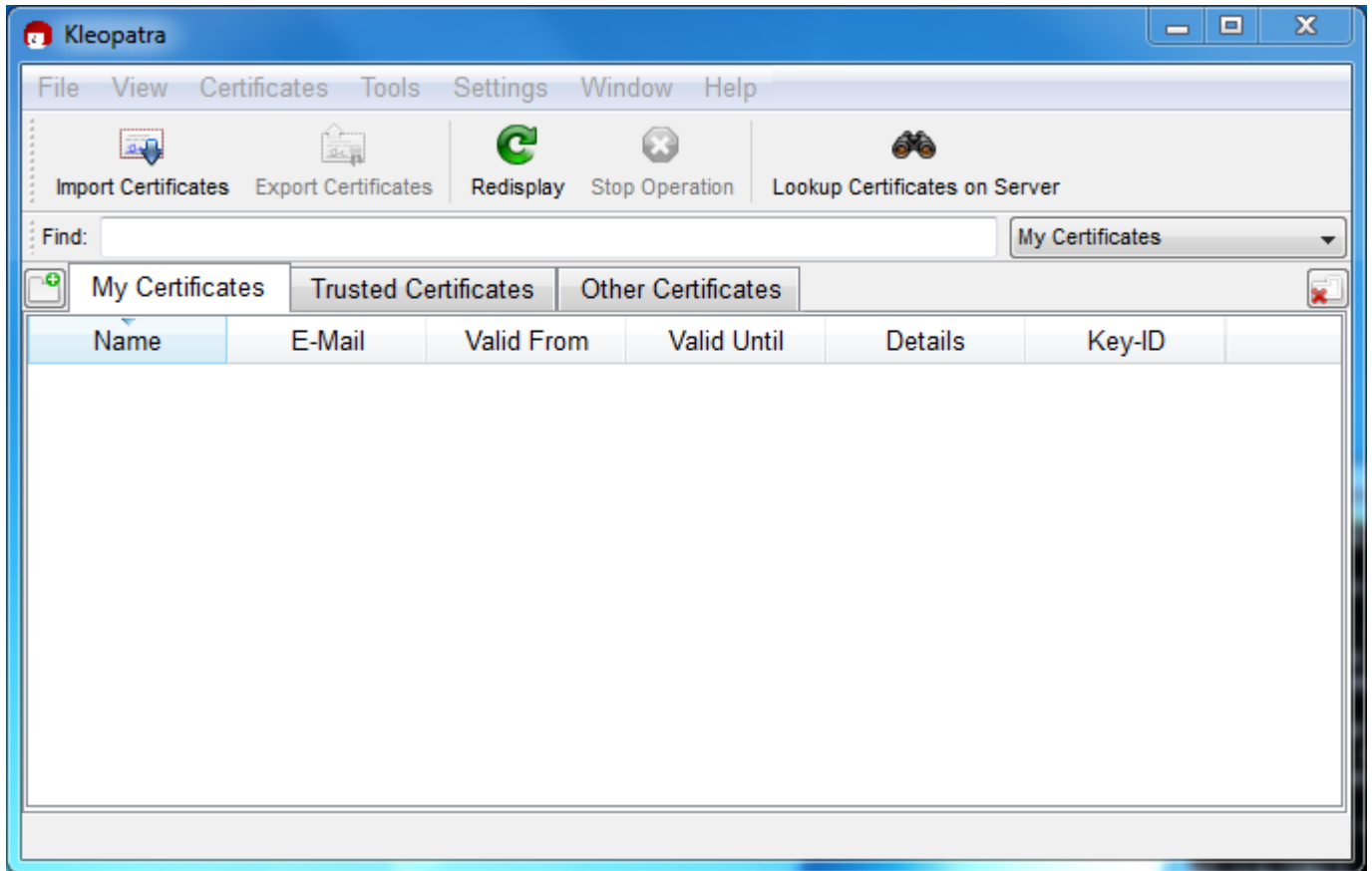
2.3.3 Signature Checking Using Gpg4win for Windows

The [Section 2.3.2, “Signature Checking Using GnuPG”](#) section describes how to verify MySQL downloads using GPG. That guide also applies to Microsoft Windows, but another option is to use a GUI tool like

[Gpg4win](#). You may use a different tool but our examples are based on Gpg4win, and utilize its bundled [Kleopatra](#) GUI.

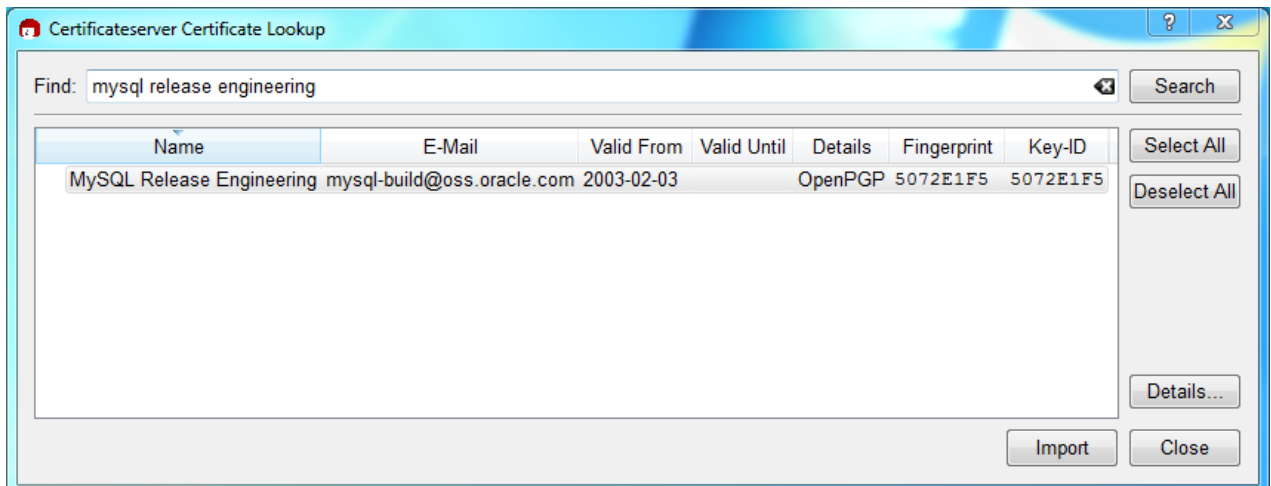
Download and install Gpg4win, and then load Kleopatra. The dialog should look similar to:

Figure 2.1 Initial screen after loading Kleopatra



Next, add the MySQL Release Engineering certificate. Do this by clicking **File, Lookup Certificates on Server**. Type "Mysql Release Engineering" into the search box and press **Search**.

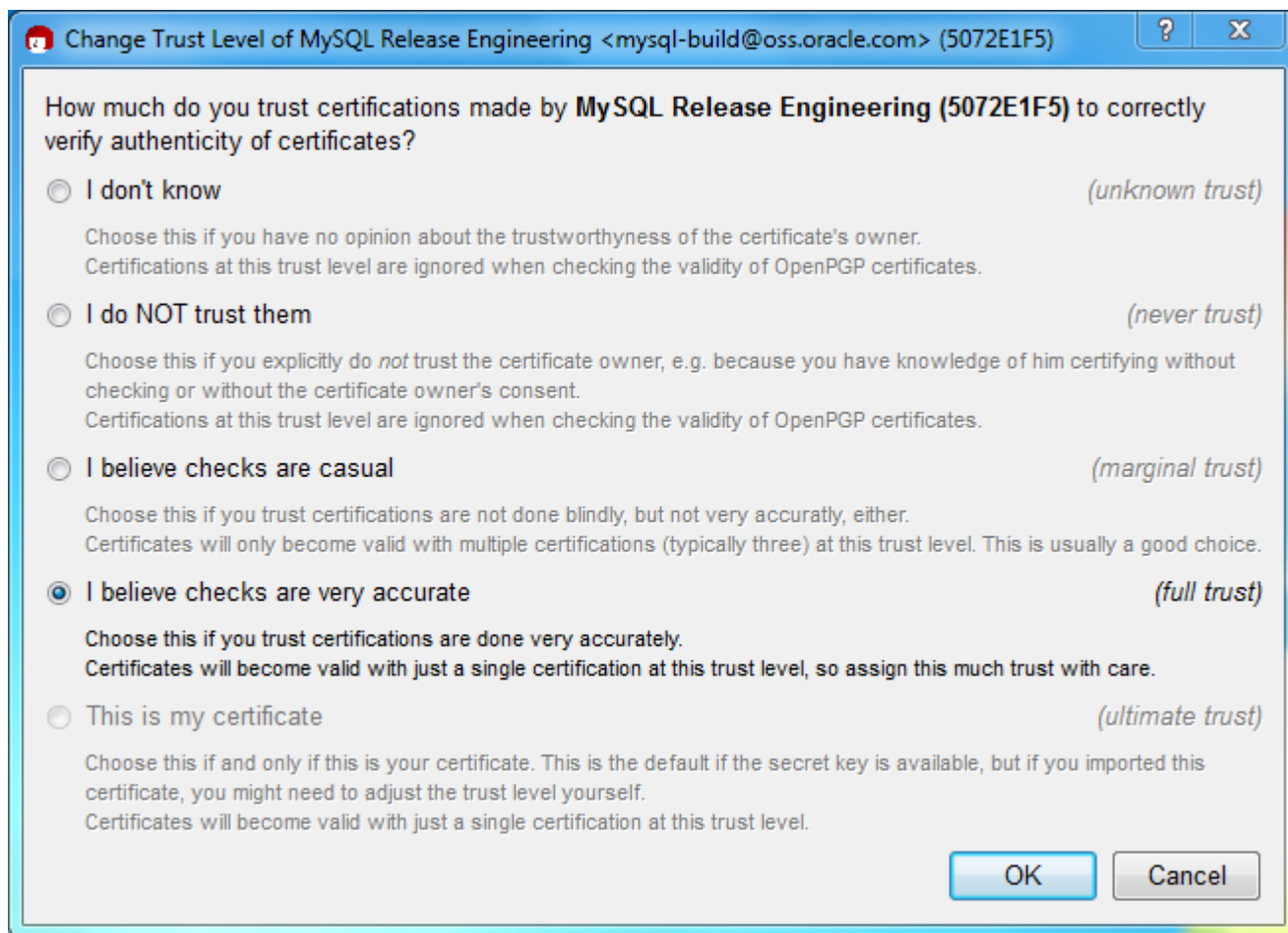
Figure 2.2 Finding the MySQL Release Engineering certificate



Select the "MySQL Release Engineering" certificate. The Fingerprint and Key-ID must be "5072E1F5", or choose **Details...** to confirm the certificate is valid. Now, import it by clicking **Import**. An import dialog will be displayed, choose **Okay**, and this certificate will now be listed under the **Imported Certificates** tab.

Next, configure the trust level for our certificate. Select our certificate, then from the main menu select **Certificates, Change Owner Trust...** We suggest choosing **I believe checks are very accurate** for our certificate, as otherwise you might not be able to verify our signature. Select **I believe checks are very accurate** and then press **OK**.

Figure 2.3 Changing the Trust level

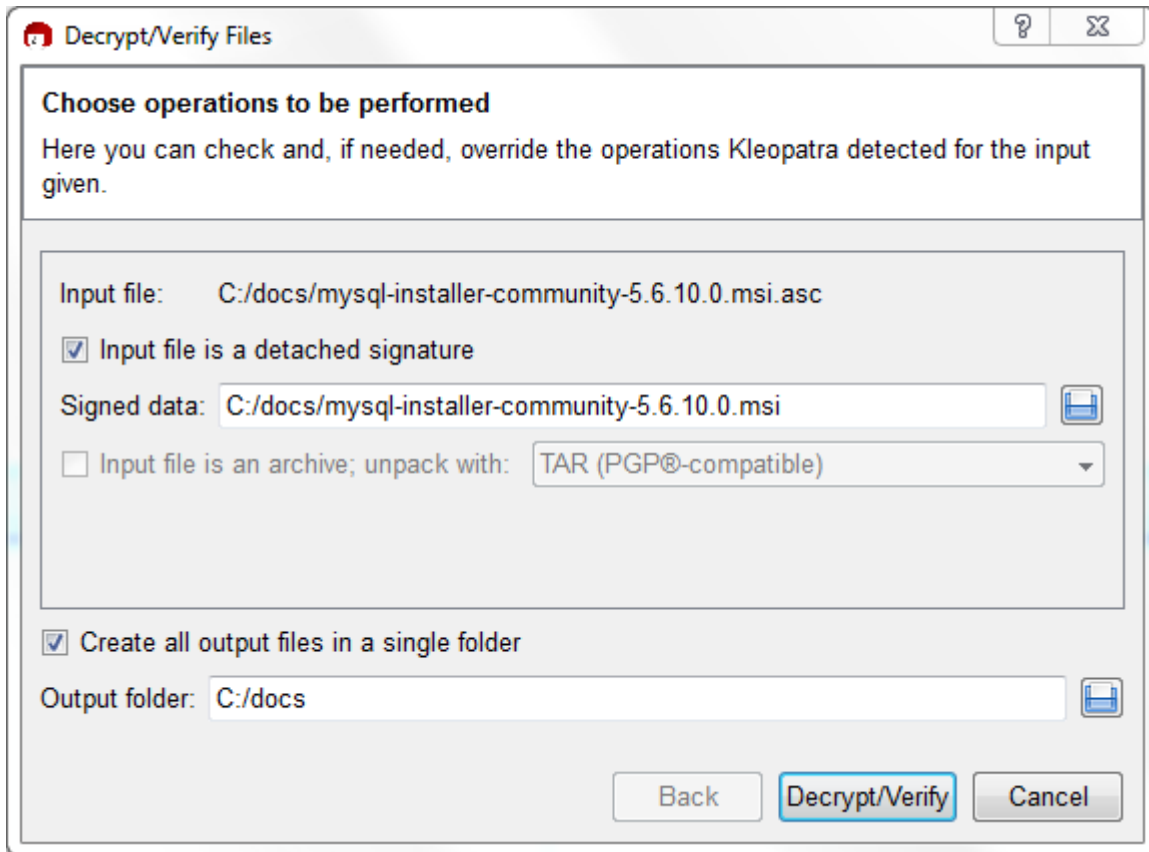


Next, verify the downloaded MySQL package file. This requires files for both the packaged file, and the signature. The signature file must have the same name as the packaged file but with an appended `.asc` extension, as shown by the example in the following table. The signature is linked to on the downloads page for each MySQL product. You must create the `.asc` file with this signature.

Table 2.2 MySQL Package and Signature Files for MySQL Installer for Microsoft Windows

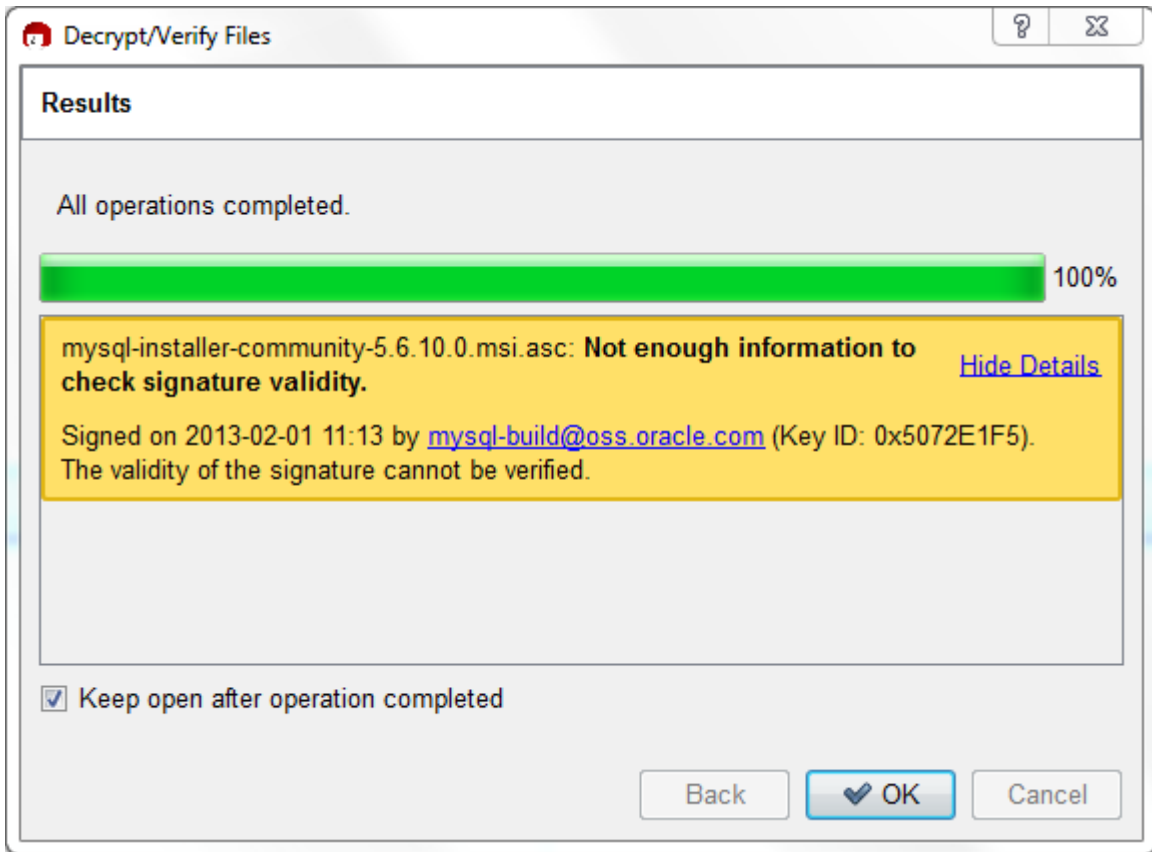
File Type	File Name
Distribution file	<code>mysql-installer-community-5.5.58.msi</code>
Signature file	<code>mysql-installer-community-5.5.58.msi.asc</code>

Make sure that both files are stored in the same directory and then run the following command to verify the signature for the distribution file. Either drag and drop the signature (`.asc`) file into Kleopatra, or load the dialog from **File, Decrypt/Verify Files...**, and then choose either the `.msi` or `.asc` file.

Figure 2.4 The Decrypt/Verify Files dialog

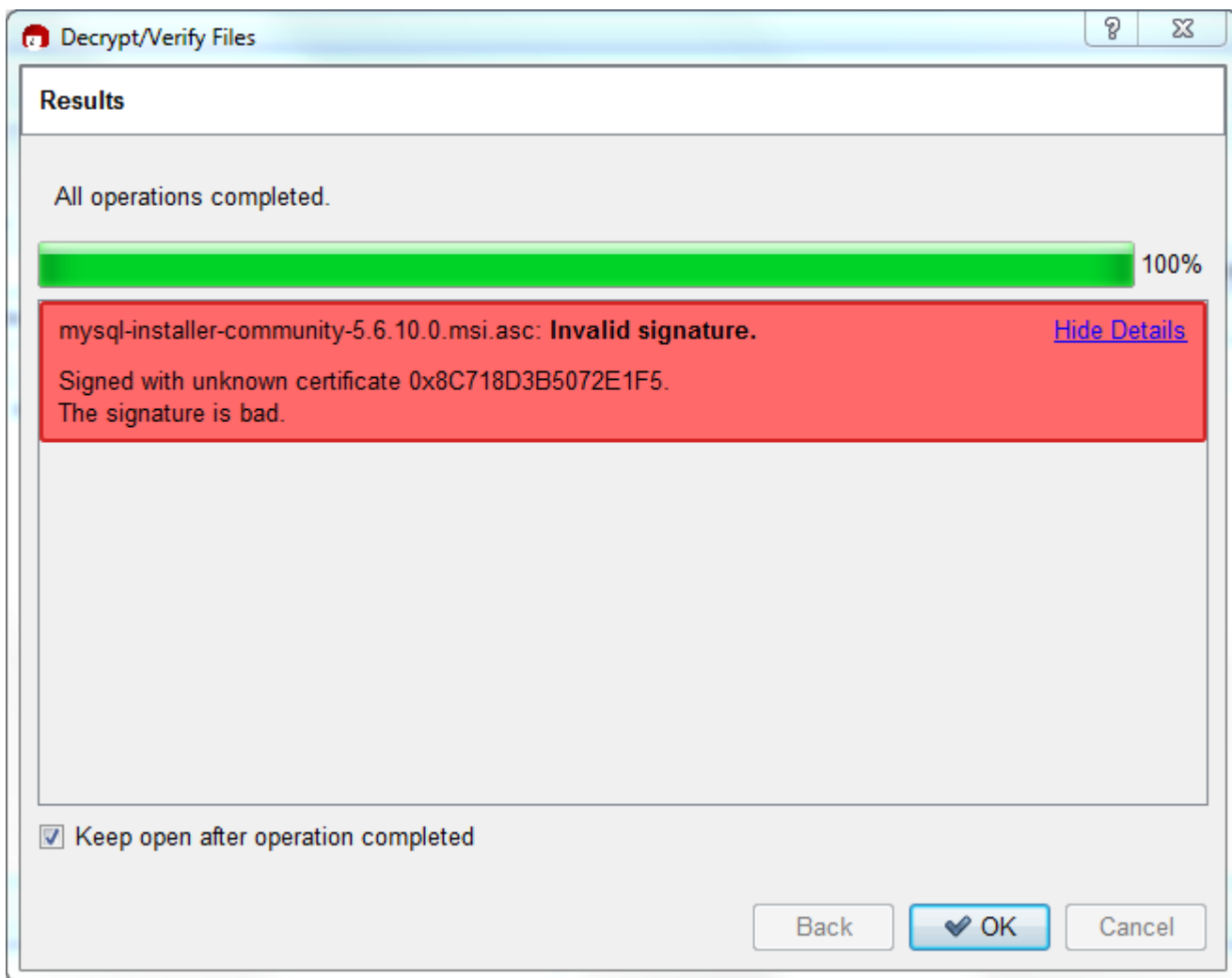
Click **Decrypt/Verify** to check the file. The two most common results will look like the following, and although the yellow warning looks problematic, the following means that the file check passed with success. You may now run this installer.

Figure 2.5 The Decrypt/Verify Results: Good



Seeing a red "The signature is bad" error means the file is invalid. Do not execute the MSI file if you see this error.

Figure 2.6 The Decrypt/Verify Results: Bad



The [Section 2.3.2, “Signature Checking Using GnuPG”](#) section explains why you probably don't see a green `Good signature` result.

2.3.4 Signature Checking Using RPM

For RPM packages, there is no separate signature. RPM packages have a built-in GPG signature and MD5 checksum. You can verify a package by running the following command:

```
shell> rpm --checksig package_name.rpm
```

Example:

```
shell> rpm --checksig MySQL-server-5.5.58-0.glibc23.i386.rpm
MySQL-server-5.5.58-0.glibc23.i386.rpm: md5 gpg OK
```

Note

If you are using RPM 4.1 and it complains about `(GPG) NOT OK (MISSING KEYS: GPG#5072e1f5)`, even though you have imported the MySQL public build

key into your own GPG keyring, you need to import the key into the RPM keyring first. RPM 4.1 no longer uses your personal GPG keyring (or GPG itself). Rather, RPM maintains a separate keyring because it is a system-wide application and a user's GPG public keyring is a user-specific file. To import the MySQL public key into the RPM keyring, first obtain the key, then use `rpm --import` to import the key. For example:

```
shell> gpg --export -a 5072e1f5 > 5072e1f5.asc  
shell> rpm --import 5072e1f5.asc
```

Alternatively, `rpm` also supports loading the key directly from a URL, and you can use this manual page:

```
shell> rpm --import http://dev.mysql.com/doc/refman/5.5/en/checking-gpg-signature.html
```

If you need to obtain the MySQL public key, see [Section 2.3.2, “Signature Checking Using GnuPG”](#).

2.4 Installation Layouts

The installation layout differs for different installation types (for example, native packages, binary tarballs, and source tarballs), which can lead to confusion when managing different systems or using different installation sources. The individual layouts are given in the corresponding installation type or platform chapter, as described following. Note that the layout of installations from vendors other than Oracle may differ from these layouts.

- [Section 5.1, “MySQL Installation Layout on Microsoft Windows”](#)
- [Section 4.1, “MySQL Layout for Source Installation”](#)
- [Table 3.1, “MySQL Installation Layout for Generic Unix/Linux Binary Package”](#)
- [Table 7.1, “MySQL Installation Layout for Linux RPM Packages”](#)
- [Table 6.1, “MySQL Installation Layout on OS X”](#)

2.5 Compiler-Specific Build Characteristics

In some cases, the compiler used to build MySQL affects the features available for use. The notes in this section apply for binary distributions provided by Oracle Corporation or that you compile yourself from source.

icc (Intel C++ Compiler) Builds

A server built with `icc` has these characteristics:

- SSL support is not included.

Chapter 3 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. For other platform-specific package formats, see the other platform-specific sections. For example, for Windows distributions, see [Chapter 5, Installing MySQL on Microsoft Windows](#).

To obtain MySQL, see [Section 2.2, “How to Get MySQL”](#).

MySQL compressed `tar` file binary distributions have names of the form `mysql-VERSION-OS.tar.gz`, where `VERSION` is a number (for example, `5.5.58`), 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-get`, 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.

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
```

If you run into problems and need to file a bug report, please use the instructions in [How to Report Bugs or Problems](#).

On Unix, 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 3.1 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

Directory	Contents of Directory
<code>docs</code>	MySQL manual in Info format
<code>man</code>	Unix manual pages
<code>include</code>	Include (header) files
<code>lib</code>	Libraries
<code>share</code>	Miscellaneous support files, including error messages, sample configuration files, SQL for database installation
<code>sql-bench</code>	Benchmarks

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 [Chapter 4, 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> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
# Next command is optional
shell> cp support-files/my-medium.cnf /etc/my.cnf
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 [Section 9.4, "Securing the Initial MySQL Accounts"](#).

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 one. 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, 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 [Section 2.2, “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. Then create a symbolic link to that directory. `tar` can uncompress and unpack the distribution if it has `z` option support:

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

The `tar` command creates a directory named `mysql-VERSION-OS`. The `ln` command makes a symbolic link to that directory. This enables you to refer more easily to the installation directory as `/usr/local/mysql`.

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 -
```

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 [Chapter 9, *Postinstallation Setup and Testing*](#).

Chapter 4 Installing MySQL from Source

Table of Contents

4.1 MySQL Layout for Source Installation	27
4.2 Installing MySQL Using a Standard Source Distribution	27
4.3 Installing MySQL Using a Development Source Tree	31
4.4 MySQL Source-Configuration Options	33
4.5 Dealing with Problems Compiling MySQL	46
4.6 MySQL Configuration and Third-Party Tools	48

Building MySQL from the source code enables you to customize build parameters, compiler optimizations, and installation location. For a list of systems on which MySQL is known to run, see <http://www.mysql.com/support/supportedplatforms/database.html>.

Before you proceed with an installation from source, check whether Oracle produces a precompiled binary distribution for your platform and whether it works for you. We put a great deal of effort into ensuring that our binaries are built with the best possible options for optimal performance. Instructions for installing binary distributions are available in [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#).

Warning

Building MySQL with nonstandard options may lead to reduced functionality, performance, or security.

Note

This section describes how to build MySQL from source using [CMake](#). Before MySQL 5.5, source builds used the GNU autotools on Unix-like systems. Source builds on Windows used [CMake](#), but the process was different from that described here. For source-building instructions for older versions of MySQL, see the *MySQL 5.1 Reference Manual*. If you are familiar with autotools but not [CMake](#), you might find these transition instructions helpful: [Autotools to CMake Transition Guide](#)

Source Installation Methods

There are two methods for installing MySQL from source:

- Use a standard MySQL source distribution. To obtain a standard distribution, see [Section 2.2, “How to Get MySQL”](#). For instructions on building from a standard distribution, see [Section 4.2, “Installing MySQL Using a Standard Source Distribution”](#).

Standard distributions are available as compressed `tar` files, Zip archives, or RPM packages. Distribution files have names of the form `mysql-VERSION.tar.gz`, `mysql-VERSION.zip`, or `mysql-VERSION.rpm`, where `VERSION` is a number like `5.5.58`. File names for source distributions can be distinguished from those for precompiled binary distributions in that source distribution names are generic and include no platform name, whereas binary distribution names include a platform name indicating the type of system for which the distribution is intended (for example, `pc-linux-i686` or `winx64`).

- Use a MySQL development tree. For information on building from one of the development trees, see [Section 4.3, “Installing MySQL Using a Development Source Tree”](#).

Source Installation System Requirements

Installation of MySQL from source requires several development tools. Some of these tools are needed no matter whether you use a standard source distribution or a development source tree. Other tool requirements depend on which installation method you use.

To install MySQL from source, the following system requirements must be satisfied, regardless of installation method:

- **CMake**, which is used as the build framework on all platforms. **CMake** can be downloaded from <http://www.cmake.org>.
- A good **make** program. Although some platforms come with their own **make** implementations, it is highly recommended that you use GNU **make** 3.75 or higher. It may already be available on your system as **gmake**. GNU **make** is available from <http://www.gnu.org/software/make/>.
- A working ANSI C++ compiler. GCC 4.2.1 or later, Sun Studio 12 or later, Visual Studio 2008 or later, and many current vendor-supplied compilers are known to work.
- Sufficient free memory. If you encounter problems such as “internal compiler error” when compiling large source files, it may be that you have too little memory. If compiling on a virtual machine, try increasing the memory allocation.
- Perl is needed if you intend to run test scripts. Most Unix-like systems include Perl. On Windows, you can use a version such as ActiveState Perl.

To install MySQL from a standard source distribution, one of the following tools is required to unpack the distribution file:

- For a **.tar.gz** compressed **tar** file: 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/>.

- For a **.zip** Zip archive: **WinZip** or another tool that can read **.zip** files.
- For an **.rpm** RPM package: The **rpmbuild** program used to build the distribution unpacks it.

To install MySQL from a development source tree, the following additional tools are required:

- The Git revision control system is required to obtain the development source code. The [GitHub Help](#) provides instructions for downloading and installing Git on different platforms. MySQL officially joined GitHub in September, 2014. For more information about MySQL's move to GitHub, refer to the announcement on the MySQL Release Engineering blog: [MySQL on GitHub](#)
- **bison** 2.1 or higher, available from <http://www.gnu.org/software/bison/>. (Version 1 is no longer supported.) Use the latest version of **bison** where possible; if you experience problems, upgrade to a later version, rather than revert to an earlier one.

bison is available from <http://www.gnu.org/software/bison/>. **bison** for Windows can be downloaded from <http://gnuwin32.sourceforge.net/packages/bison.htm>. Download the package labeled “Complete package, excluding sources”. On Windows, the default location for **bison** is the `C:\Program Files\GnuWin32` directory. Some utilities may fail to find **bison** because of the space in the directory name.

Also, Visual Studio may simply hang if there are spaces in the path. You can resolve these problems by installing into a directory that does not contain a space; for example `C:\GnuWin32`.

- On Solaris Express, `m4` must be installed in addition to `bison`. `m4` is available from <http://www.gnu.org/software/m4/>.

Note

If you have to install any programs, modify your `PATH` environment variable to include any directories in which the programs are located. See [Setting Environment Variables](#).

If you run into problems and need to file a bug report, please use the instructions in [How to Report Bugs or Problems](#).

4.1 MySQL Layout for Source Installation

By default, when you install MySQL after compiling it from source, the installation step installs files under `/usr/local/mysql`. The component locations under the installation directory are the same as for binary distributions. See [Table 3.1, “MySQL Installation Layout for Generic Unix/Linux Binary Package”](#), and [Section 5.1, “MySQL Installation Layout on Microsoft Windows”](#). To configure installation locations different from the defaults, use the options described at [Section 4.4, “MySQL Source-Configuration Options”](#).

4.2 Installing MySQL Using a Standard Source Distribution

To install MySQL from a standard source distribution:

1. Verify that your system satisfies the tool requirements listed at [Chapter 4, *Installing MySQL from Source*](#).
2. Obtain a distribution file using the instructions in [Section 2.2, “How to Get MySQL”](#).
3. Configure, build, and install the distribution using the instructions in this section.
4. Perform postinstallation procedures using the instructions in [Chapter 9, *Postinstallation Setup and Testing*](#).

In MySQL 5.5, `CMake` is used as the build framework on all platforms. The instructions given here should enable you to produce a working installation. For additional information on using `CMake` to build MySQL, see [How to Build MySQL Server with CMake](#).

If you start from a source RPM, use the following command to make a binary RPM that you can install. If you do not have `rpmbuild`, use `rpm` instead.

```
shell> rpmbuild --rebuild --clean MySQL-VERSION.src.rpm
```

The result is one or more binary RPM packages that you install as indicated in [Section 7.1, “Installing MySQL on Linux Using RPM Packages”](#).

The sequence for installation from a compressed `tar` file or Zip archive source distribution is similar to the process for installing from a generic binary distribution (see [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#)), except that it is used on all platforms and includes steps to configure and compile the distribution. For example, with a compressed `tar` file source distribution on Unix, the basic installation command sequence looks like this:

```
# Preconfiguration setup
```

```

shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
# Beginning of source-build specific instructions
shell> tar zxvf mysql-VERSION.tar.gz
shell> cd mysql-VERSION
shell> mkdir bld
shell> cd bld
shell> cmake ..
shell> make
shell> make install
# End of source-build specific instructions
# Postinstallation setup
shell> cd /usr/local/mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
# Next command is optional
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server

```

A more detailed version of the source-build specific instructions is shown following.

Note

The procedure shown here does not set up any passwords for MySQL accounts. After following the procedure, proceed to [Chapter 9, Postinstallation Setup and Testing](#), for postinstallation setup and testing.

Perform Preconfiguration Setup

On Unix, set up the `mysql` user and group that will be used to run and execute the MySQL server and own the database directory. For details, see [Creating a `mysql` System User and Group](#), in [Chapter 3, Installing MySQL on Unix/Linux Using Generic Binaries](#). Then perform the following steps as the `mysql` user, except as noted.

Obtain and Unpack the Distribution

Pick the directory under which you want to unpack the distribution and change location into it.

Obtain a distribution file using the instructions in [Section 2.2, “How to Get MySQL”](#).

Unpack the distribution into the current directory:

- To unpack a compressed `tar` file, `tar` can uncompress and unpack the distribution if it has `z` option support:

```
shell> tar zxvf mysql-VERSION.tar.gz
```

If your `tar` does not have `z` option support, use `gunzip` to unpack the distribution and `tar` to unpack it:

```
shell> gunzip < mysql-VERSION.tar.gz | tar xvf -
```

Alternatively, `CMake` can uncompress and unpack the distribution:

```
shell> cmake -E tar zxvf mysql-VERSION.tar.gz
```

- To unpack a Zip archive, use [WinZip](#) or another tool that can read `.zip` files.

Unpacking the distribution file creates a directory named `mysql-VERSION`.

Configure the Distribution

Change location into the top-level directory of the unpacked distribution:

```
shell> cd mysql-VERSION
```

Build out of the source tree to keep the tree clean. If the top-level source directory is named `mysql-src` under your current working directory, you can build in a directory named `bld` at the same level. Create the directory and go there:

```
shell> mkdir bld
shell> cd bld
```

Configure the build directory. The minimum configuration command includes no options to override configuration defaults:

```
shell> cmake ../mysql-src
```

The build directory needs not be outside the source tree. For example, you can build in a directory named `bld` under the top-level source tree. To do this, starting with `mysql-src` as your current working directory, create the directory `bld` and then go there:

```
shell> mkdir bld
shell> cd bld
```

Configure the build directory. The minimum configuration command includes no options to override configuration defaults:

```
shell> cmake ..
```

If you have multiple source trees at the same level (for example, to build multiple versions of MySQL), the second strategy can be advantageous. The first strategy places all build directories at the same level, which requires that you choose a unique name for each. With the second strategy, you can use the same name for the build directory within each source tree. The following instructions assume this second strategy.

On Windows, specify the development environment. For example, the following commands configure MySQL for 32-bit or 64-bit builds, respectively:

```
shell> cmake .. -G "Visual Studio 9 2008"
shell> cmake .. -G "Visual Studio 9 2008 Win64"
```

On OS X, to use the Xcode IDE:

```
shell> cmake .. -G Xcode
```

When you run `cmake`, you might want to add options to the command line. Here are some examples:

- `-DBUILD_CONFIG=mysql_release`: Configure the source with the same build options used by Oracle to produce binary distributions for official MySQL releases.

- `-DCMAKE_INSTALL_PREFIX=dir_name`: Configure the distribution for installation under a particular location.
- `-DCPACK_MONOLITHIC_INSTALL=1`: Cause `make package` to generate a single installation file rather than multiple files.
- `-DWITH_DEBUG=1`: Build the distribution with debugging support.

For a more extensive list of options, see [Section 4.4, “MySQL Source-Configuration Options”](#).

To list the configuration options, use one of the following commands:

```
shell> cmake .. -L # overview
shell> cmake .. -LH # overview with help text
shell> cmake .. -LAH # all params with help text
shell> ccmake .. # interactive display
```

If `CMake` fails, you might need to reconfigure by running it again with different options. If you do reconfigure, take note of the following:

- If `CMake` is run after it has previously been run, it may use information that was gathered during its previous invocation. This information is stored in `CMakeCache.txt`. When `CMake` starts up, it looks for that file and reads its contents if it exists, on the assumption that the information is still correct. That assumption is invalid when you reconfigure.
- Each time you run `CMake`, you must run `make` again to recompile. However, you may want to remove old object files from previous builds first because they were compiled using different configuration options.

To prevent old object files or configuration information from being used, run these commands in the build directory on Unix before re-running `CMake`:

```
shell> make clean
shell> rm CMakeCache.txt
```

Or, on Windows:

```
shell> devenv MySQL.sln /clean
shell> del CMakeCache.txt
```

If you are going to send mail to a MySQL mailing list to ask for configuration assistance, first check the files in the `CMakeFiles` directory for useful information about the failure. To file a bug report, please use the instructions in [How to Report Bugs or Problems](#).

Build the Distribution

On Unix:

```
shell> make
shell> make VERBOSE=1
```

The second command sets `VERBOSE` to show the commands for each compiled source.

Use `gmake` instead on systems where you are using GNU `make` and it has been installed as `gmake`.

On Windows:

```
shell> devenv MySQL.sln /build RelWithDebInfo
```

If you have gotten to the compilation stage, but the distribution does not build, see [Section 4.5, “Dealing with Problems Compiling MySQL”](#), for help. If that does not solve the problem, please enter it into our bugs database using the instructions given in [How to Report Bugs or Problems](#). If you have installed the latest versions of the required tools, and they crash trying to process our configuration files, please report that also. However, if you get a `command not found` error or a similar problem for required tools, do not report it. Instead, make sure that all the required tools are installed and that your `PATH` variable is set correctly so that your shell can find them.

Install the Distribution

On Unix:

```
shell> make install
```

This installs the files under the configured installation directory (by default, `/usr/local/mysql`). You might need to run the command as `root`.

To install in a specific directory, add a `DESTDIR` parameter to the command line:

```
shell> make install DESTDIR="/opt/mysql"
```

Alternatively, generate installation package files that you can install where you like:

```
shell> make package
```

This operation produces one or more `.tar.gz` files that can be installed like generic binary distribution packages. See [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#). If you run `CMake` with `-DCPACK_MONOLITHIC_INSTALL=1`, the operation produces a single file. Otherwise, it produces multiple files.

On Windows, generate the data directory, then create a `.zip` archive installation package:

```
shell> devenv MySQL.sln /build RelWithDebInfo /project initial_database
shell> devenv MySQL.sln /build RelWithDebInfo /project package
```

You can install the resulting `.zip` archive where you like. See [Section 5.7, “Installing MySQL on Microsoft Windows Using a noinstall Zip Archive”](#).

Perform Postinstallation Setup

The remainder of the installation process involves setting up the configuration file, creating the core databases, and starting the MySQL server. For instructions, see [Chapter 9, *Postinstallation Setup and Testing*](#).

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 [Chapter 9, *Postinstallation Setup and Testing*](#).

4.3 Installing MySQL Using a Development Source Tree

This section describes how to install MySQL from the latest development source code, which is hosted on [GitHub](#). To obtain the MySQL Server source code from this repository hosting service, you can set up a local MySQL Git repository.

On [GitHub](#), MySQL Server and other MySQL projects are found on the [MySQL](#) page. The MySQL Server project is a single repository that contains branches for several MySQL series.

MySQL officially joined GitHub in September, 2014. For more information about MySQL's move to GitHub, refer to the announcement on the MySQL Release Engineering blog: [MySQL on GitHub](#)

Prerequisites for Installing from Development Source

To install MySQL from a development source tree, your system must satisfy the tool requirements outlined in [Chapter 4, Installing MySQL from Source](#).

Setting Up a MySQL Git Repository

To set up a MySQL Git repository on your machine, use this procedure:

1. Clone the MySQL Git repository to your machine. The following command clones the MySQL Git repository to a directory named `mysql-server`. The initial download will take some time to complete, depending on the speed of your connection.

```
~$ git clone https://github.com/mysql/mysql-server.git
Cloning into 'mysql-server'...
remote: Counting objects: 1035465, done.
remote: Total 1035465 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1035465/1035465), 437.48 MiB | 5.10 MiB/s, done.
Resolving deltas: 100% (855607/855607), done.
Checking connectivity... done.
Checking out files: 100% (21902/21902), done.
```

2. When the clone operation completes, the contents of your local MySQL Git repository appear similar to the following:

```
~$ cd mysql-server
~/mysql-server$ ls
BUILD          COPYING          libmysqld       regex           tests
BUILD-CMAKE    debug            libservices     scripts         unittest
client         Docs             man             sql            VERSION
cmake          extra            mysql-test      sql-bench      vio
CMakeLists.txt include          mysys           sql-common     win
cmd-line-utils INSTALL-SOURCE   packaging       storage         zlib
config.h.cmake INSTALL-WIN-SOURCE plugin           strings
configure.cmake libmysql         README          support-files
```

3. Use the `git branch -r` command to view the remote tracking branches for the MySQL repository.

```
~/mysql-server$ git branch -r
origin/5.5
origin/5.6
origin/5.7
origin/HEAD -> origin/5.7
origin/cluster-7.2
origin/cluster-7.3
origin/cluster-7.4
```

4. To view the branches that are checked out in your local repository, issue the `git branch` command. When you cloned the MySQL Git repository, the MySQL 5.7 branch was checked out automatically. The asterisk identifies the 5.7 branch as the active branch.

```
~/mysql-server$ git branch
* 5.7
```

5. To check out a different MySQL branch, run the `git checkout` command, specifying the branch name. For example, to checkout the MySQL 5.5 branch:

```
~/mysql-server$ git checkout 5.5
Branch 5.5 set up to track remote branch 5.5 from origin.
Switched to a new branch '5.5'
```

6. Run `git branch` again to verify that the MySQL 5.5 branch is present. MySQL 5.5, which is the last branch you checked out, is marked by an asterisk indicating that it is the active branch.

```
~/mysql-server$ git branch
* 5.5
  5.7
```

The `git checkout` command is also used to switch branches. For example, to make MySQL 5.7 the active branch again, you would run `git checkout 5.7`.

7. To obtain changes made after your initial setup of the MySQL Git repository, switch to the branch you want to update and issue the `git pull` command:

```
~/mysql-server$ git checkout 5.5
~/mysql-server$ git pull
```

To examine the commit history, use the `git log` option:

```
~/mysql-server$ git log
```

You can also browse commit history and source code on the GitHub [MySQL](#) site.

If you see changes or code that you have a question about, send an email to the MySQL [internals](#) mailing list. See [MySQL Mailing Lists](#). For information about contributing a patch, see [Contributing to MySQL Server](#).

8. After you have cloned the MySQL Git repository and have checked out the branch you want to build, you can build MySQL Server from the source code. Instructions are provided in [Section 4.2, “Installing MySQL Using a Standard Source Distribution”](#), except that you skip the part about obtaining and unpacking the distribution.

Be careful about installing a build from a distribution source tree on a production machine. The installation command may overwrite your live release installation. If you already have MySQL installed and do not want to overwrite it, run `CMake` with values for the `CMAKE_INSTALL_PREFIX`, `MYSQL_TCP_PORT`, and `MYSQL_UNIX_ADDR` options different from those used by your production server. For additional information about preventing multiple servers from interfering with each other, see [Running Multiple MySQL Instances on One Machine](#).

Play hard with your new installation. For example, try to make new features crash. Start by running `make test`. See [The MySQL Test Suite](#).

4.4 MySQL Source-Configuration Options

The `CMake` program provides a great deal of control over how you configure a MySQL source distribution. Typically, you do this using options on the `CMake` command line. For information about options supported by `CMake`, run either of these commands in the top-level source directory:

```
shell> cmake . -LH
shell> cmake .
```

You can also affect `CMake` using certain environment variables. See [Chapter 11, MySQL Program Environment Variables](#).

The following table shows the available `CMake` options. In the `Default` column, `PREFIX` stands for the value of the `CMAKE_INSTALL_PREFIX` option, which specifies the installation base directory. This value is used as the parent location for several of the installation subdirectories.

Table 4.1 MySQL Source-Configuration Option Reference (CMake)

Formats	Description	Default	Introduced	Removed
<code>BUILD_CONFIG</code>	Use same build options as official releases		5.5.7	
<code>CMAKE_BUILD_TYPE</code>	Type of build to produce	<code>RelWithDebInfo</code>	5.5.7	
<code>CMAKE_CXX_FLAGS</code>	Flags for C++ Compiler			
<code>CMAKE_C_FLAGS</code>	Flags for C Compiler			
<code>CMAKE_INSTALL_PREFIX</code>	Installation base directory	<code>/usr/local/mysql</code>	5.5.8	
<code>COMPILATION_COMMENT</code>	Comment about compilation environment		5.5.7	
<code>CPACK_MONOLITHIC_INSTALL</code>	Whether package build produces single file	<code>OFF</code>	5.5.7	
<code>DEFAULT_CHARSET</code>	The default server character set	<code>latin1</code>	5.5.7	
<code>DEFAULT_COLLATION</code>	The default server collation	<code>latin1_swedish_ci</code>	5.5.7	
<code>ENABLED_LOCAL_INFILE</code>	Whether to enable LOCAL for LOAD DATA INFILE	<code>OFF</code>	5.5.7	
<code>ENABLED_PROFILING</code>	Whether to enable query profiling code	<code>ON</code>	5.5.7	
<code>ENABLE_DEBUG_SYNC</code>	Whether to enable Debug Sync support	<code>ON</code>	5.5.7	5.5.55
<code>ENABLE_DOWNLOADS</code>	Whether to download optional files	<code>OFF</code>	5.5.7	
<code>ENABLE_DTRACE</code>	Whether to include DTrace support		5.5.7	
<code>ENABLE_GCOV</code>	Whether to include gcov support		5.5.14	
<code>IGNORE_AIO_CHECK</code>	With <code>-DBUILD_CONFIG=mysql_release</code> , ignore libaio check	<code>OFF</code>	5.5.9	
<code>INSTALL_BINDIR</code>	User executables directory	<code>PREFIX/bin</code>	5.5.7	
<code>INSTALL_DOCDIR</code>	Documentation directory	<code>PREFIX/docs</code>	5.5.7	
<code>INSTALL_DOCREADMEDIR</code>	README file directory	<code>PREFIX</code>	5.5.7	
<code>INSTALL_INCLUDEDIR</code>	Header file directory	<code>PREFIX/include</code>	5.5.7	
<code>INSTALL_INFODIR</code>	Info file directory	<code>PREFIX/docs</code>	5.5.7	
<code>INSTALL_LAYOUT</code>	Select predefined installation layout	<code>STANDALONE</code>	5.5.7	
<code>INSTALL_LIBDIR</code>	Library file directory	<code>PREFIX/lib</code>	5.5.7	
<code>INSTALL_MANDIR</code>	Manual page directory	<code>PREFIX/man</code>	5.5.7	

MySQL Source-Configuration Options

Formats	Description	Default	Introduced	Removed
INSTALL_MYSQLSHAREDIR	Shared data directory	PREFIX/share	5.5.7	
INSTALL_MYSQLTESTDIR	mysql-test directory	PREFIX/mysql-test	5.5.7	
INSTALL_PLUGINDIR	Plugin directory	PREFIX/lib/plugin	5.5.7	
INSTALL_SBINDIR	Server executable directory	PREFIX/bin	5.5.7	
INSTALL_SCRIPTDIR	Scripts directory	PREFIX/scripts	5.5.7	
INSTALL_SECURE_FILE_PRIV	secure_file_priv default value	platform specific	5.5.53	
INSTALL_SECURE_FILE_PRIV	secure_file_priv default value for libmysqld		5.5.53	
INSTALL_SHAREDIR	aclocal/mysql.m4 installation directory	PREFIX/share	5.5.7	
INSTALL_SQLBENCHDIR	sql-bench directory	PREFIX	5.5.7	
INSTALL_SUPPORTFILES	Extra support files directory	PREFIX/support-files	5.5.7	
MEMCACHED_HOME	Path to memcached	[none]	5.5.16-ndb-7.2.2	
MYSQL_DATADIR	Data directory		5.5.7	
MYSQL_MAINTAINER_MODE	Whether to enable MySQL maintainer-specific development environment	OFF	5.5.7	
MYSQL_PROJECT_NAME	Windows/OS X project name	MySQL	5.5.21	
MYSQL_TCP_PORT	TCP/IP port number	3306	5.5.7	
MYSQL_UNIX_ADDR	Unix socket file	/tmp/mysql.sock	5.5.7	
ODBC_INCLUDES	ODBC includes directory			
ODBC_LIB_DIR	ODBC library directory			
SYSCONFDIR	Option file directory		5.5.7	
TMPDIR	tmpdir default value		5.5.36	
WITHOUT_SERVER	Do not build the server	OFF		
WITHOUT_XXX_STORAGE_ENGINE	Exclude storage engine xxx from build		5.5.7	
WITH_ASAN	Enable AddressSanitizer	OFF	5.5.35	
WITH_BUNDLED_LIBEVENT	Use bundled libevent when building ndbmemcache	ON	5.5.16-ndb-7.2.2	
WITH_BUNDLED_MEMCACHED	Use bundled memcached when building ndbmemcache	ON	5.5.16-ndb-7.2.2	
WITH_CLASSPATH	Classpath to use when building MySQL Cluster Connector for Java. Default is an empty string.			

Formats	Description	Default	Introduced	Removed
WITH_DEBUG	Whether to include debugging support	OFF	5.5.7	
WITH_EMBEDDED_SERVER	Whether to build embedded server	OFF	5.5.7	
WITH_EMBEDDED_SHARED_LIBRARY	Whether to build a shared embedded server library	OFF	5.5.37	
WITH_ERROR_INSERT	Enable error injection in the NDB storage engine. Should not be used for building binaries intended for production.	OFF		
WITH_EXTRA_CHARSETS	Which extra character sets to include	all	5.5.7	
WITH_LIBEDIT	Use bundled libedit library	ON	5.5.7	
WITH_LIBWRAP	Whether to include libwrap (TCP wrappers) support	OFF	5.5.7	
WITH_NDBCLUSTER	Build the NDB storage engine; alias for WITH_NDBCLUSTER_STORAGE_ENGINE	ON		
WITH_NDBCLUSTER_STORAGE_ENGINE	Build the NDB storage engine	ON		
WITH_NDBMTD	Build multi-threaded data node.	ON		
WITH_NDB_BINLOG	Enable binary logging by default by mysqld.	ON		
WITH_NDB_DEBUG	Produce a debug build for testing or troubleshooting.	OFF		
WITH_NDB_JAVA	Enable building of Java and ClusterJ support. Enabled by default. Supported in MySQL Cluster only.	ON	5.5.27-ndb-7.2.9	
WITH_NDB_PORT	Default port used by a management server built with this option. If this option was not used to build it, the management server's default port is 1186.	[none]		
WITH_NDB_TEST	Include NDB API test programs.	OFF		
WITH_READLINE	Use bundled readline library	OFF	5.5.7	
WITH_SSL	Type of SSL support	bundled	5.5.7	
WITH_UNIT_TESTS	Compile MySQL with unit tests	ON		
WITH_UNIXODBC	Enable unixODBC support	OFF		
WITH_VALGRIND	Whether to compile in Valgrind header files	OFF	5.5.6	

Formats	Description	Default	Introduced	Removed
<code>WITH_ZLIB</code>	Type of zlib support	<code>bundled</code>	5.5.7	
<code>WITH_XXX_STORAGE_ENGINE</code>	Compile storage engine xxx statically into server		5.5.7	

The following sections provide more information about `CMake` options.

- [General Options](#)
- [Installation Layout Options](#)
- [Storage Engine Options](#)
- [Feature Options](#)
- [Compiler Flags](#)
- [CMake Options for Compiling NDB Cluster](#)

For boolean options, the value may be specified as 1 or `ON` to enable the option, or as 0 or `OFF` to disable the option.

Many options configure compile-time defaults that can be overridden at server startup. For example, the `CMAKE_INSTALL_PREFIX`, `MYSQL_TCP_PORT`, and `MYSQL_UNIX_ADDR` options that configure the default installation base directory location, TCP/IP port number, and Unix socket file can be changed at server startup with the `--basedir`, `--port`, and `--socket` options for `mysqld`. Where applicable, configuration option descriptions indicate the corresponding `mysqld` startup option.

General Options

- `-DBUILD_CONFIG=mysql_release`

This option configures a source distribution with the same build options used by Oracle to produce binary distributions for official MySQL releases.

- `-DCMAKE_BUILD_TYPE=type`

The type of build to produce:

- `RelWithDebInfo`: Enable optimizations and generate debugging information. This is the default MySQL build type.
- `Debug`: Disable optimizations and generate debugging information. This build type is also used if the `WITH_DEBUG` option is enabled. That is, `-DWITH_DEBUG=1` has the same effect as `-DCMAKE_BUILD_TYPE=Debug`.
- `-DCPACK_MONOLITHIC_INSTALL=bool`

This option affects whether the `make package` operation produces multiple installation package files or a single file. If disabled, the operation produces multiple installation package files, which may be useful if you want to install only a subset of a full MySQL installation. If enabled, it produces a single file for installing everything.

Installation Layout Options

The `CMAKE_INSTALL_PREFIX` option indicates the base installation directory. Other options with names of the form `INSTALL_xxx` that indicate component locations are interpreted relative to the prefix and their values are relative pathnames. Their values should not include the prefix.

- `-DCMAKE_INSTALL_PREFIX=dir_name`

The installation base directory.

This value can be set at server startup with the `--basedir` option.

- `-DINSTALL_BINDIR=dir_name`

Where to install user programs.

- `-DINSTALL_DOCDIR=dir_name`

Where to install documentation.

- `-DINSTALL_DOCREADMEDIR=dir_name`

Where to install `README` files.

- `-DINSTALL_INCLUDEDIR=dir_name`

Where to install header files.

- `-DINSTALL_INFODIR=dir_name`

Where to install Info files.

- `-DINSTALL_LAYOUT=name`

Select a predefined installation layout:

- `STANDALONE`: Same layout as used for `.tar.gz` and `.zip` packages. This is the default.
- `RPM`: Layout similar to RPM packages.
- `SVR4`: Solaris package layout.
- `DEB`: DEB package layout (experimental).

You can select a predefined layout but modify individual component installation locations by specifying other options. For example:

```
shell> cmake . -DINSTALL_LAYOUT=SVR4 -DMYSQL_DATADIR=/var/mysql/data
```

- `-DINSTALL_LIBDIR=dir_name`

Where to install library files.

- `-DINSTALL_MANDIR=dir_name`

Where to install manual pages.

- `-DINSTALL_MYSQLSHAREDIR=dir_name`

Where to install shared data files.

- `-DINSTALL_MYSQLTESTDIR=dir_name`

Where to install the `mysql-test` directory. As of MySQL 5.5.32, to suppress installation of this directory, explicitly set the option to the empty value (`-DINSTALL_MYSQLTESTDIR=`).

- `-DINSTALL_PLUGINDIR=dir_name`

The location of the plugin directory.

This value can be set at server startup with the `--plugin_dir` option.

- `-DINSTALL_SBINDIR=dir_name`

Where to install the `mysqld` server.

- `-DINSTALL_SCRIPTDIR=dir_name`

Where to install `mysql_install_db`.

- `-DINSTALL_SECURE_FILE_PRIVDIR=dir_name`

The default value for the `secure_file_priv` system variable. The default value is platform specific and depends on the value of the `INSTALL_LAYOUT` CMake option; see the description of the `secure_file_priv` system variable in [Server System Variables](#).

This option was added in MySQL 5.5.53. To set the value for the `libmysqld` embedded server, use `INSTALL_SECURE_FILE_PRIV_EMBEDDEDIR`.

- `-DINSTALL_SECURE_FILE_PRIV_EMBEDDEDIR=dir_name`

The default value for the `secure_file_priv` system variable, for the `libmysqld` embedded server. This option was added in MySQL 5.5.53.

- `-DINSTALL_SHAREDIR=dir_name`

Where to install `aclocal/mysql.m4`.

- `-DINSTALL_SQLBENCHDIR=dir_name`

Where to install the `sql-bench` directory. To suppress installation of this directory, explicitly set the option to the empty value (`-DINSTALL_SQLBENCHDIR=`).

- `-DINSTALL_SUPPORTFILESDIR=dir_name`

Where to install extra support files.

- `-DMYSQL_DATADIR=dir_name`

The location of the MySQL data directory.

This value can be set at server startup with the `--datadir` option.

- `-DODBC_INCLUDES=dir_name`

The location of the ODBC includes directory, and may be used while configuring Connector/ODBC.

- `-DODBC_LIB_DIR=dir_name`

The location of the ODBC library directory, and may be used while configuring Connector/ODBC.

- `-DSYSCONFDIR=dir_name`

The default `my.cnf` option file directory.

This location cannot be set at server startup, but you can start the server with a given option file using the `--defaults-file=file_name` option, where `file_name` is the full path name to the file.

- `-DTMPDIR=dir_name`

The default location to use for the `tmpdir` system variable. If unspecified, the value defaults to `P_tmpdir` in `<stdio.h>`. This option was added in MySQL 5.6.16.

Storage Engine Options

Storage engines are built as plugins. You can build a plugin as a static module (compiled into the server) or a dynamic module (built as a dynamic library that must be installed into the server using the `INSTALL PLUGIN` statement or the `--plugin-load` option before it can be used). Some plugins might not support static or dynamic building.

The `InnoDB`, `MyISAM`, `MERGE`, `MEMORY`, and `CSV` engines are mandatory (always compiled into the server) and need not be installed explicitly.

To compile a storage engine statically into the server, use `-DWITH_engine_STORAGE_ENGINE=1`. Some permissible `engine` values are `ARCHIVE`, `BLACKHOLE`, `EXAMPLE`, `FEDERATED`, `NDBCLUSTER` (`NDB`), `PARTITION` (partitioning support), and `PERFSHEMA` (Performance Schema). Examples:

```
-DWITH_ARCHIVE_STORAGE_ENGINE=1
-DWITH_BLACKHOLE_STORAGE_ENGINE=1
-DWITH_PERFSHEMA_STORAGE_ENGINE=1
```

Note

`WITH_NDBCLUSTER_STORAGE_ENGINE` is supported only when building NDB Cluster using the NDB Cluster sources. It cannot be used to enable clustering support in other MySQL source trees or distributions. In MySQL NDB Cluster 7.2 source distributions, it is enabled by default. See [Building NDB Cluster from Source on Linux](#), and [Compiling and Installing NDB Cluster from Source on Windows](#), for more information.

To exclude a storage engine from the build, use `-DWITHOUT_engine_STORAGE_ENGINE=1`. Examples:

```
-DWITHOUT_EXAMPLE_STORAGE_ENGINE=1
-DWITHOUT_FEDERATED_STORAGE_ENGINE=1
-DWITHOUT_PARTITION_STORAGE_ENGINE=1
```

If neither `-DWITH_engine_STORAGE_ENGINE` nor `-DWITHOUT_engine_STORAGE_ENGINE` are specified for a given storage engine, the engine is built as a shared module, or excluded if it cannot be built as a shared module.

Feature Options

- `-DCOMPILATION_COMMENT=string`

A descriptive comment about the compilation environment.

- `-DDEFAULT_CHARSET=charset_name`

The server character set. By default, MySQL uses the `latin1` (cp1252 West European) character set.

`charset_name` may be one of `binary`, `armscii8`, `ascii`, `big5`, `cp1250`, `cp1251`, `cp1256`, `cp1257`, `cp850`, `cp852`, `cp866`, `cp932`, `dec8`, `eucjpms`, `euckr`, `gb2312`, `gbk`, `geostd8`, `greek`, `hebrew`, `hp8`, `keybcs2`, `koi8r`, `koi8u`, `latin1`, `latin2`, `latin5`, `latin7`, `macce`, `macroman`, `sjis`, `swe7`, `tis620`, `ucs2`, `ujis`, `utf8`, `utf8mb4`, `utf16`, `utf32`. The permissible character sets are listed in the `cmake/character_sets.cmake` file as the value of `CHARSETS_AVAILABLE`.

This value can be set at server startup with the `--character_set_server` option.

- `-DDEFAULT_COLLATION=collation_name`

The server collation. By default, MySQL uses `latin1_swedish_ci`. Use the `SHOW COLLATION` statement to determine which collations are available for each character set.

This value can be set at server startup with the `--collation_server` option.

- `-DENABLE_DEBUG_SYNC=bool`

Note

As of MySQL 5.5.55, `ENABLE_DEBUG_SYNC` is removed and enabling `WITH_DEBUG` enables Debug Sync.

Whether to compile the Debug Sync facility into the server. This facility is used for testing and debugging. This option is enabled by default, but has no effect unless MySQL is configured with debugging enabled. If debugging is enabled and you want to disable Debug Sync, use `-DENABLE_DEBUG_SYNC=0`.

When compiled in, Debug Sync is disabled by default at runtime. To enable it, start `mysqld` with the `--debug-sync-timeout=N` option, where `N` is a timeout value greater than 0. (The default value is 0, which disables Debug Sync.) `N` becomes the default timeout for individual synchronization points.

For a description of the Debug Sync facility and how to use synchronization points, see [MySQL Internals: Test Synchronization](#).

- `-DENABLE_DOWNLOADS=bool`

Whether to download optional files. For example, with this option enabled, `CMake` downloads the Google Test distribution that is used by the test suite to run unit tests.

- `-DENABLE_DTRACE=bool`

Whether to include support for DTrace probes. For information about DTrace, see [Tracing mysqld Using DTrace](#)

- `-DENABLE_GCOV=bool`

Whether to include gcov support (Linux only).

- `-DENABLED_LOCAL_INFILE=bool`

This option controls the compiled-in default `LOCAL` capability for the MySQL client library. Clients that make no explicit arrangements therefore have `LOCAL` capability disabled or enabled according to the `ENABLED_LOCAL_INFILE` setting specified at MySQL build time.

By default, the client library in MySQL binary distributions is compiled with `ENABLED_LOCAL_INFILE` enabled. If you compile MySQL from source, configure it with `ENABLED_LOCAL_INFILE` disabled or enabled based on whether clients that make no explicit arrangements should have `LOCAL` capability disabled or enabled, respectively.

`ENABLED_LOCAL_INFILE` controls the default for client-side `LOCAL` capability. For the server, the `local_infile` system variable controls server-side `LOCAL` capability. To explicitly cause the server to refuse or permit `LOAD DATA LOCAL` statements (regardless of how client programs and libraries are configured at build time or runtime), start `mysqld` with `local_infile` disabled or enabled, respectively. `local_infile` can also be set at runtime. See [Security Issues with LOAD DATA LOCAL](#).

- `-DENABLED_PROFILING=bool`

Whether to enable query profiling code (for the `SHOW PROFILE` and `SHOW PROFILES` statements).

- `-DIGNORE_AIO_CHECK=bool`

If the `-DBUILD_CONFIG=mysql_release` option is given on Linux, the `libaio` library must be linked in by default. If you do not have `libaio` or do not want to install it, you can suppress the check for it by specifying `-DIGNORE_AIO_CHECK=1`. This option was added in MySQL 5.5.9.

- `-DMYSQL_MAINTAINER_MODE=bool`

Whether to enable a MySQL maintainer-specific development environment. If enabled, this option causes compiler warnings to become errors.

- `-DMYSQL_PROJECT_NAME=name`

For Windows or OS X, the project name to incorporate into the project file name. This option was added in MySQL 5.5.21.

- `-DMYSQL_TCP_PORT=port_num`

The port number on which the server listens for TCP/IP connections. The default is 3306.

This value can be set at server startup with the `--port` option.

- `-DMYSQL_UNIX_ADDR=file_name`

The Unix socket file path on which the server listens for socket connections. This must be an absolute path name. The default is `/tmp/mysql.sock`.

This value can be set at server startup with the `--socket` option.

- `-DWITH_ASAN=bool`

Whether to enable AddressSanitizer, for compilers that support it. The default is off. This option was added in MySQL 5.5.35.

- `-DWITH_DEBUG=bool`

Whether to include debugging support.

Configuring MySQL with debugging support enables you to use the `--debug="d,parser_debug"` option when you start the server. This causes the Bison parser that is used to process SQL statements to dump a parser trace to the server's standard error output. Typically, this output is written to the error log.

As of MySQL 5.5.55, enabling `WITH_DEBUG` also enables Debug Sync. For a description of the Debug Sync facility and how to use synchronization points, see [MySQL Internals: Test Synchronization](#).

- `-DWITH_EMBEDDED_SERVER=bool`

Whether to build the `libmysqld` embedded server library.

- `-DWITH_EMBEDDED_SHARED_LIBRARY=bool`

Whether to build a shared `libmysqld` embedded server library. This option was added in MySQL 5.5.37.

- `-DWITH_EXTRA_CHARSETS=name`

Which extra character sets to include:

- `all`: All character sets. This is the default.
- `complex`: Complex character sets.
- `none`: No extra character sets.

- `-DWITH_LIBEDIT=bool`

Whether to use the `libedit` library bundled with the distribution.

- `-DWITH_LIBWRAP=bool`

Whether to include `libwrap` (TCP wrappers) support.

- `-DWITH_READLINE=bool`

Whether to use the `readline` library bundled with the distribution.

- `-DWITH_SSL=ssl_type`

The type of SSL support to include, if any:

- `no`: No SSL support. This is the default before MySQL 5.5.56. As of 5.5.56, this is no longer a permitted value and the default is `bundled`.
- `yes`: Use the system SSL library if present, else the library bundled with the distribution.
- `bundled`: Use the SSL library bundled with the distribution. This is the default as of MySQL 5.5.56.
- `system`: Use the system SSL library.

For information about using SSL support, see [Using Secure Connections](#).

- `-DWITH_UNIT_TESTS={ON|OFF}`

If enabled, compile MySQL with unit tests. The default is ON unless the server is not being compiled.

- `-DWITH_UNIXODBC=1`

Enables unixODBC support, for Connector/ODBC.

- `-DWITH_VALGRIND=bool`

Whether to compile in the Valgrind header files, which exposes the Valgrind API to MySQL code. The default is `OFF`.

To generate a Valgrind-aware debug build, `-DWITH_VALGRIND=1` normally is combined with `-DWITH_DEBUG=1`. See [Building Debug Configurations](#).

- `-DWITH_ZLIB=zlib_type`

Some features require that the server be built with compression library support, such as the `COMPRESS()` and `UNCOMPRESS()` functions, and compression of the client/server protocol. The `WITH_ZLIB` indicates the source of `zlib` support:

- `bundled`: Use the `zlib` library bundled with the distribution. This is the default.
- `system`: Use the system `zlib` library.
- `-DWITHOUT_SERVER=bool`

Whether to build without the MySQL server. The default is `OFF`, which does build the server.

Compiler Flags

- `-DCMAKE_C_FLAGS="flags"`

Flags for the C Compiler.

- `-DCMAKE_CXX_FLAGS="flags"`

Flags for the C++ Compiler.

To specify your own C and C++ compiler flags, for flags that do not affect optimization, use the `CMAKE_C_FLAGS` and `CMAKE_CXX_FLAGS` CMake options.

When providing your own compiler flags, you might want to specify `CMAKE_BUILD_TYPE` as well.

For example, to create a 32-bit release build on a 64-bit Linux machine, do this:

```
shell> mkdir bld
shell> cd bld
shell> cmake .. -DCMAKE_C_FLAGS=-m32 \
             -DCMAKE_CXX_FLAGS=-m32 \
             -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

If you set flags that affect optimization (`-Onumber`), you must set the `CMAKE_C_FLAGS_build_type` and/or `CMAKE_CXX_FLAGS_build_type` options, where `build_type` corresponds to the `CMAKE_BUILD_TYPE` value. To specify a different optimization for the default build type (`RelWithDebInfo`) set the `CMAKE_C_FLAGS_RELWITHDEBINFO` and `CMAKE_CXX_FLAGS_RELWITHDEBINFO` options. For example, to compile on Linux with `-O3` and with debug symbols, do this:

```
shell> cmake .. -DCMAKE_C_FLAGS_RELWITHDEBINFO="-O3 -g" \
             -DCMAKE_CXX_FLAGS_RELWITHDEBINFO="-O3 -g"
```

CMake Options for Compiling NDB Cluster

The following options are for use when building MySQL NDB Cluster 7.2 or later. These options are supported only with the MySQL NDB Cluster 7.2 and later NDB Cluster sources; they are not supported when using sources from the MySQL 5.5 Server tree.

- `-DMEMCACHED_HOME=dir_name`

Perform the build using the memcached (version 1.6 or later) installed in the system directory indicated by `dir_name`. Files from this installation that are used in the build include the memcached binary, header files, and libraries, as well as the `memcached_utilities` library and the header file `engine_testapp.h`.

You must leave this option unset when building `ndbmemcache` using the bundled memcached sources (`WITH_BUNDLED_MEMCACHED` option); in other words, the bundled sources are used by default).

This option was added in MySQL NDB Cluster 7.2.2.

While additional CMake options—such as for SASL authorization and for providing `dtrace` support—are available for use when compiling `memcached` from external sources, these options are currently not enabled for the `memcached` sources bundled with NDB Cluster.

- `-DWITH_BUNDLED_LIBEVENT={ON|OFF}`

Use the `libevent` included in the NDB Cluster sources when building NDB Cluster with `ndbmemcached` support (MySQL NDB Cluster 7.2.2 and later). Enabled by default. OFF causes the system's `libevent` to be used instead.

- `-DWITH_BUNDLED_MEMCACHED={ON|OFF}`

Build the memcached sources included in the NDB Cluster source tree (MySQL NDB Cluster 7.2.3 and later), then use the resulting memcached server when building the `ndbmemcache` engine. In this case, `make install` places the `memcached` binary in the installation `bin` directory, and the `ndbmemcache` engine shared library file `ndb_engine.so` in the installation `lib` directory.

This option is ON by default.

- `-DWITH_CLASSPATH=path`

Sets the classpath for building NDB Cluster Connector for Java. The default is empty. In MySQL NDB Cluster 7.2.9 and later, this option is ignored if `-DWITH_NDB_JAVA=OFF` is used.

- `-DWITH_ERROR_INSERT={ON|OFF}`

Enables error injection in the NDB kernel. For testing only; not intended for use in building production binaries. The default is OFF.

- `-DWITH_NDBCLUSTER_STORAGE_ENGINE={ON|OFF}`

Build and link in support for the NDB (NDBCLUSTER) storage engine in `mysqld`. The default is ON.

- `-DWITH_NDBCLUSTER={ON|OFF}`

This is an alias for `WITH_NDBCLUSTER_STORAGE_ENGINE`.

- `-DWITH_NDBMTD={ON|OFF}`

Build the multi-threaded data node executable `ndbmt.d`. The default is ON.

- `-DWITH_NDB_BINLOG={ON|OFF}`

Enable binary logging by default in the `mysqld` built using this option. ON by default.

- `-DWITH_NDB_DEBUG={ON|OFF}`

Enable building the debug versions of the NDB Cluster binaries. OFF by default.

- `-DWITH_NDB_JAVA={ON|OFF}`

Enable building NDB Cluster with Java support, including `ClusterJ`.

This option was added in MySQL NDB Cluster 7.2.9, and is ON by default. If you do not wish to compile NDB Cluster with Java support, you must disable it explicitly by specifying `-DWITH_NDB_JAVA=OFF` when running `CMake`. Otherwise, if Java cannot be found, configuration of the build fails.

- `-DWITH_NDB_PORT=port`

Causes the NDB Cluster management server (`ndb_mgmd`) that is built to use this `port` by default. If this option is unset, the resulting management server tries to use port 1186 by default.

- `-DWITH_NDB_TEST={ON|OFF}`

If enabled, include a set of NDB API test programs. The default is OFF.

4.5 Dealing with Problems Compiling MySQL

The solution to many problems involves reconfiguring. If you do reconfigure, take note of the following:

- If `CMake` is run after it has previously been run, it may use information that was gathered during its previous invocation. This information is stored in `CMakeCache.txt`. When `CMake` starts up, it looks for that file and reads its contents if it exists, on the assumption that the information is still correct. That assumption is invalid when you reconfigure.
- Each time you run `CMake`, you must run `make` again to recompile. However, you may want to remove old object files from previous builds first because they were compiled using different configuration options.

To prevent old object files or configuration information from being used, run the following commands before re-running `CMake`:

On Unix:

```
shell> make clean
shell> rm CMakeCache.txt
```

On Windows:

```
shell> devenv MySQL.sln /clean
shell> del CMakeCache.txt
```

If you build outside of the source tree, remove and recreate your build directory before re-running `CMake`. For instructions on building outside of the source tree, see [How to Build MySQL Server with CMake](#).

On some systems, warnings may occur due to differences in system include files. The following list describes other problems that have been found to occur most often when compiling MySQL:

- To define which C and C++ compilers to use, you can define the `CC` and `CXX` environment variables. For example:

```
shell> CC=gcc
shell> CXX=g++
shell> export CC CXX
```

To specify your own C and C++ compiler flags, use the `CMAKE_C_FLAGS` and `CMAKE_CXX_FLAGS` CMake options. See [Compiler Flags](#).

To see what flags you might need to specify, invoke `mysql_config` with the `--cflags` option.

- To see what commands are executed during the compile stage, after using CMake to configure MySQL, run `make VERBOSE=1` rather than just `make`.
- If compilation fails, check whether the `MYSQL_MAINTAINER_MODE` option is enabled. This mode causes compiler warnings to become errors, so disabling it may enable compilation to proceed.
- If your compile fails with errors such as any of the following, you must upgrade your version of `make` to GNU `make`:

```
make: Fatal error in reader: Makefile, line 18:
Badly formed macro assignment
```

Or:

```
make: file `Makefile' line 18: Must be a separator (:
```

Or:

```
pthread.h: No such file or directory
```

Solaris and FreeBSD are known to have troublesome `make` programs.

GNU `make` 3.75 is known to work.

- The `sql_yacc.cc` file is generated from `sql_yacc.yy`. Normally, the build process does not need to create `sql_yacc.cc` because MySQL comes with a pregenerated copy. However, if you do need to re-create it, you might encounter this error:

```
"sql_yacc.yy", line xxx fatal: default action causes potential...
```

This is a sign that your version of `yacc` is deficient. You probably need to install a recent version of `bison` (the GNU version of `yacc`) and use that instead.

Versions of `bison` older than 1.75 may report this error:

```
sql_yacc.yy:#####: fatal error: maximum table size (32767) exceeded
```

The maximum table size is not actually exceeded; the error is caused by bugs in older versions of `bison`.

For information about acquiring or updating tools, see the system requirements in [Chapter 4, Installing MySQL from Source](#).

4.6 MySQL Configuration and Third-Party Tools

Third-party tools that need to determine the MySQL version from the MySQL source can read the `VERSION` file in the top-level source directory. The file lists the pieces of the version separately. For example, if the version is MySQL 5.7.4-m14, the file looks like this:

```
MYSQL_VERSION_MAJOR=5
MYSQL_VERSION_MINOR=7
MYSQL_VERSION_PATCH=4
MYSQL_VERSION_EXTRA=-m14
```

If the source is not for a General Availability (GA) release, the `MYSQL_VERSION_EXTRA` value will be nonempty. For the example, the value corresponds to Milestone 14.

To construct a five-digit number from the version components, use this formula:

```
MYSQL_VERSION_MAJOR*10000 + MYSQL_VERSION_MINOR*100 + MYSQL_VERSION_PATCH
```

Chapter 5 Installing MySQL on Microsoft Windows

Table of Contents

5.1 MySQL Installation Layout on Microsoft Windows	52
5.2 Choosing An Installation Package	53
5.3 MySQL Installer for Windows	54
5.3.1 MySQL Installer Initial Setup	55
5.3.2 Installation Workflow	59
5.3.3 MySQL Installer Product Catalog and Dashboard	66
5.3.4 MySQLInstallerConsole Reference	69
5.4 MySQL Notifier	73
5.4.1 MySQL Notifier Usage	73
5.4.2 Setting Up Remote Monitoring in MySQL Notifier	80
5.5 Installing MySQL on Microsoft Windows Using an MSI Package	85
5.5.1 Using the MySQL Installation Wizard	87
5.5.2 Automating MySQL Installation on Microsoft Windows Using the MSI Package	90
5.5.3 Removing MySQL When Installed from the MSI Package	91
5.6 MySQL Server Instance Configuration Wizard	92
5.6.1 Starting the MySQL Server Instance Configuration Wizard	92
5.6.2 Choosing a Maintenance Option	93
5.6.3 Choosing a Configuration Type	93
5.6.4 The Server Type Dialog	94
5.6.5 The Database Usage Dialog	96
5.6.6 The InnoDB Tablespace Dialog	97
5.6.7 The Concurrent Connections Dialog	98
5.6.8 The Networking and Strict Mode Options Dialog	99
5.6.9 The Character Set Dialog	100
5.6.10 The Service Options Dialog	100
5.6.11 The Security Options Dialog	101
5.6.12 The Confirmation Dialog	103
5.6.13 MySQL Server Instance Config Wizard: Creating an Instance from the Command Line	104
5.7 Installing MySQL on Microsoft Windows Using a noinstall Zip Archive	107
5.7.1 Extracting the Install Archive	107
5.7.2 Creating an Option File	108
5.7.3 Selecting a MySQL Server Type	109
5.7.4 Starting the Server for the First Time	109
5.7.5 Starting MySQL from the Windows Command Line	111
5.7.6 Customizing the PATH for MySQL Tools	111
5.7.7 Starting MySQL as a Windows Service	112
5.7.8 Testing The MySQL Installation	115
5.8 Troubleshooting a Microsoft Windows MySQL Server Installation	116
5.9 Windows Postinstallation Procedures	117
5.10 Upgrading MySQL on Windows	119

Important

MySQL Community 5.5 Server requires the Microsoft Visual C++ 2008 Redistributable Package to run on Windows platforms. Users should make sure the package has been installed on the system before installing the server. The package is available at the [Microsoft Download Center](#).

MySQL is available for Microsoft Windows, for both 32-bit and 64-bit versions. For supported Windows platform information, see <http://www.mysql.com/support/supportedplatforms/database.html>.

There are different methods to install MySQL on Microsoft Windows.

MySQL Installer Method

The simplest and recommended method is to download MySQL Installer (for Windows) and let it install and configure all of the MySQL products on your system. Here is how:

1. Download MySQL Installer from <http://dev.mysql.com/downloads/installer/> and execute it.

Note

Unlike the standard MySQL Installer, the smaller "web-community" version does not bundle any MySQL applications but it will download the MySQL products you choose to install.

2. Choose the appropriate **Setup Type** for your system. Typically you will choose **Developer Default** to install MySQL server and other MySQL tools related to MySQL development, helpful tools like MySQL Workbench. Or, choose the **Custom** setup type to manually select your desired MySQL products.

Note

Multiple versions of MySQL server can exist on a single system. You can choose one or multiple versions.

3. Complete the installation process by following the MySQL Installation wizard's instructions. This will install several MySQL products and start the MySQL server.

MySQL is now installed. If you configured MySQL as a service, then Windows will automatically start MySQL server every time you restart your system.

Note

You probably also installed other helpful MySQL products like MySQL Workbench and MySQL Notifier on your system. Consider loading [MySQL Workbench](#) to check your new MySQL server connection, and [Section 5.4, "MySQL Notifier"](#) to view the connection's status. By default, these two programs automatically start after installing MySQL.

This process also installs the MySQL Installer application on your system, and later you can use MySQL Installer to upgrade or reconfigure your MySQL products.

Additional Installation Information

It is possible to run MySQL as a standard application or as a Windows service. By using a service, you can monitor and control the operation of the server through the standard Windows service management tools. For more information, see [Section 5.7.7, "Starting MySQL as a Windows Service"](#).

Generally, you should install MySQL on Windows using an account that has administrator rights. Otherwise, you may encounter problems with certain operations such as editing the `PATH` environment variable or accessing the [Service Control Manager](#). Once installed, MySQL does not need to be executed using a user with Administrator privileges.

For a list of limitations on the use of MySQL on the Windows platform, see [Windows Platform Limitations](#).

In addition to the MySQL Server package, you may need or want additional components to use MySQL with your application or development environment. These include, but are not limited to:

- To connect to the MySQL server using ODBC, you must have a Connector/ODBC driver. For more information, including installation and configuration instructions, see [MySQL Connector/ODBC Developer Guide](#).

Note

MySQL Installer will install and configure Connector/ODBC for you.

- To use MySQL server with .NET applications, you must have the Connector/Net driver. For more information, including installation and configuration instructions, see [MySQL Connector/Net Developer Guide](#).

Note

MySQL Installer will install and configure MySQL Connector/Net for you.

MySQL distributions for Windows can be downloaded from <http://dev.mysql.com/downloads/>. See [Section 2.2, “How to Get MySQL”](#).

MySQL for Windows is available in several distribution formats, detailed here. Generally speaking, you should use MySQL Installer. It contains more features and MySQL products than the older MSI, is simpler to use than the Zip file, and you need no additional tools to get MySQL up and running. MySQL Installer automatically installs MySQL Server and additional MySQL products, creates an options file, starts the server, and enables you to create default user accounts. For more information on choosing a package, see [Section 5.2, “Choosing An Installation Package”](#).

- Binary installer distributions. There are two different installable distributions that come packaged as a Microsoft Windows Installer (MSI) package that you can install manually or automatically on your systems. The preferred MySQL Installer package includes MySQL Server and additional MySQL products including MySQL Workbench, MySQL Notifier, and MySQL for Excel. MySQL Installer can also be used to upgrade these product in the future. The older MSI package contains all the files you need to install and configure MySQL server, but no additional components.

For instructions on installing MySQL using MySQL Installer, see [Section 5.3, “MySQL Installer for Windows”](#).

- The standard binary distribution (packaged as a Zip file) contains all of the necessary files that you unpack into your chosen location. This package contains all of the files in the full Windows MSI Installer package, but does not include an installation program.

For instructions on installing MySQL using the Zip file, see [Section 5.7, “Installing MySQL on Microsoft Windows Using a noinstall Zip Archive”](#).

- The source distribution format contains all the code and support files for building the executables using the Visual Studio compiler system.

For instructions on building MySQL from source on Windows, see [Chapter 4, *Installing MySQL from Source*](#).

MySQL on Windows considerations:

- **Large Table Support**

If you need tables with a size larger than 4GB, install MySQL on an NTFS or newer file system. Do not forget to use `MAX_ROWS` and `AVG_ROW_LENGTH` when you create tables. See [CREATE TABLE Syntax](#).

- **MySQL and Virus Checking Software**

Virus-scanning software such as Norton/Symantec Anti-Virus on directories containing MySQL data and temporary tables can cause issues, both in terms of the performance of MySQL and the virus-scanning software misidentifying the contents of the files as containing spam. This is due to the fingerprinting mechanism used by the virus-scanning software, and the way in which MySQL rapidly updates different files, which may be identified as a potential security risk.

After installing MySQL Server, it is recommended that you disable virus scanning on the main directory (`datadir`) used to store your MySQL table data. There is usually a system built into the virus-scanning software to enable specific directories to be ignored.

In addition, by default, MySQL creates temporary files in the standard Windows temporary directory. To prevent the temporary files also being scanned, configure a separate temporary directory for MySQL temporary files and add this directory to the virus scanning exclusion list. To do this, add a configuration option for the `tmpdir` parameter to your `my.ini` configuration file. For more information, see [Section 5.7.2, “Creating an Option File”](#).

5.1 MySQL Installation Layout on Microsoft Windows

For MySQL 5.5 on Windows, the default installation directory is `C:\Program Files\MySQL\MySQL Server 5.5`. Some Windows users prefer to install in `C:\mysql`, the directory that formerly was used as the default. However, the layout of the subdirectories remains the same.

All of the files are located within this parent directory, using the structure shown in the following table.

Table 5.1 Default MySQL Installation Layout for Microsoft Windows

Directory	Contents of Directory	Notes
<code>bin, scripts</code>	<code>mysqld</code> server, client and utility programs	
<code>%ALLUSERSPROFILE%\MySQL\MySQL Server 5.5\</code>	Log files, databases (Windows XP, Windows Server 2003)	The Windows system variable <code>%ALLUSERSPROFILE%</code> defaults to <code>C:\Documents and Settings\All Users\Application Data</code>
<code>%PROGRAMDATA%\MySQL\MySQL Server 5.5\</code>	Log files, databases (Vista, Windows 7, Windows Server 2008, and newer)	The Windows system variable <code>%PROGRAMDATA%</code> defaults to <code>C:\ProgramData</code>
<code>examples</code>	Example programs and scripts	
<code>include</code>	Include (header) files	
<code>lib</code>	Libraries	
<code>share</code>	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation	

If you install MySQL using a Windows MSI package, this package creates and sets up the data directory that the installed server will use, but as of MySQL 5.5.5, it also creates a pristine “template” data directory named `data` under the installation directory. This directory can be useful when the machine will be used to run multiple instances of MySQL: After an installation has been performed using an MSI package, the

template data directory can be copied to set up additional MySQL instances. See [Running Multiple MySQL Instances on One Machine](#).

5.2 Choosing An Installation Package

For MySQL 5.5, there are multiple installation package formats to choose from when installing MySQL on Windows.

Note

MySQL Installer and the "Complete Package" methods for installing MySQL are similar, but different. The MySQL Installer is the newer and more advanced option, and it includes all functionality found within the "Complete Package."

Note

Program Database (PDB) files (with file name extension `pdb`) provide information for debugging your MySQL installation in the event of a problem. These files are included in ZIP Archive distributions (but not MSI distributions) of MySQL.

- **MySQL Installer:** This package has a file name similar to `mysql-installer-community-5.5.58.0.msi` or `mysql-installer-commercial-5.5.58.0.msi`, and utilizes MSIs to automatically install MySQL server and other products. It will download and apply updates to itself, and for each of the installed products. It also configures the additional non-server products.

The installed products are configurable, and this includes: documentation with samples and examples, connectors (such as C, C++, J, NET, and ODBC), MySQL Workbench, MySQL Notifier, MySQL for Excel, and the MySQL Server with its components.

MySQL Installer will run on all Windows platforms that are supported by MySQL (see <http://www.mysql.com/support/supportedplatforms/database.html>).

Note

Because MySQL Installer is not a native component of Microsoft Windows and depends on .NET, it will not work on minimal installation options like the "Server Core" version of Windows Server 2008.

For instructions on installing MySQL using MySQL Installer, see [Section 5.3, "MySQL Installer for Windows"](#).

- **The Complete Package:** This package has a file name similar to `mysql-5.5.58-win32.msi` or `mysql-5.5.58-winx64.zip`, and contains all files needed for a complete Windows installation, including the Configuration Wizard. This package includes optional components such as the embedded server and benchmark suite.
- **The Noinstall Archive:** This package has a file name similar to `mysql-5.5.58-win32.zip` or `mysql-5.5.58-winx64.zip`, and contains all the files found in the Complete install package, with the exception of the GUI. It also contains PDB files. This package does not include an automated installer, and must be manually installed and configured.

MySQL Installer is recommended for most users.

Your choice of install package affects the installation process you must follow. If you choose to use MySQL Installer, see [Section 5.3, "MySQL Installer for Windows"](#). If you choose to install a standard MSI package, see [Section 5.5, "Installing MySQL on Microsoft Windows Using an MSI Package"](#). If you choose to

install a Noinstall archive, see [Section 5.7, “Installing MySQL on Microsoft Windows Using a noinstall Zip Archive”](#).

5.3 MySQL Installer for Windows

MySQL Installer is a standalone application designed to ease the complexity of installing and managing MySQL products that run on Microsoft Windows. It supports the following MySQL products:

- MySQL Servers

A single MySQL Installer instance can install and manage multiple MySQL server versions. For example, a single MySQL Installer instance can install (and update) versions 5.6, 5.7, and 8.0 on the same host. A host *cannot* have both community and commercial editions of MySQL server installed.

- MySQL Applications

MySQL Workbench, MySQL Shell, MySQL Router, MySQL for Visual Studio, MySQL for Excel, MySQL Notifier, and MySQL Utilities.

- MySQL Connectors

MySQL Connector/Net, MySQL Connector/Python, Connector/ODBC, MySQL Connector/J, Connector/C, and Connector/C++.

- Documentation and Samples

MySQL Reference Manuals (by version) in PDF format and MySQL database samples (by version).

MySQL Installer Community Edition

Download this edition from <http://dev.mysql.com/downloads/installer/> to install the community version of all MySQL products for Windows. Select one of the following MySQL Installer package options:

- *Web*: Contains MySQL Installer and configuration files only. The web package downloads only the MySQL products you select to install, but it requires an internet connection for each download. The size of this file is approximately 2 MB; the name of the file has the form `mysql-installer-community-web-VERSION.N.msi` where `VERSION` is the MySQL server version number such as 5.7 and `N` is the package number, which begins at 0.
- *Full*: Bundles all of the MySQL products for Windows (including the MySQL server). The file size is over 300 MB, and its name has the form `mysql-installer-community-VERSION.N.msi` where `VERSION` is the MySQL Server version number such as 5.7 and `N` is the package number, which begins at 0.

MySQL Installer Commercial Edition

Download this edition from <https://edelivery.oracle.com/> to install the commercial edition of all MySQL products for Windows. The Commercial Edition includes all of the products in the Community Edition and also includes the following products:

- Workbench SE/EE
- MySQL Enterprise Backup
- MySQL Enterprise Firewall

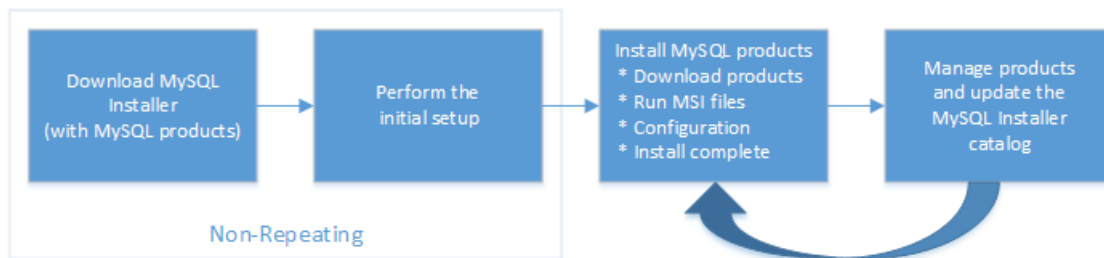
This edition integrates with your My Oracle Support (MOS) account. For knowledge-base content and patches, see [My Oracle Support](#).

5.3.1 MySQL Installer Initial Setup

- [MySQL Installer Licensing and Support Authentication](#)
- [Choosing a Setup Type](#)
- [Path Conflicts](#)
- [Check Requirements](#)
- [MySQL Installer Configuration Files](#)

When you download MySQL Installer for the first time, a setup wizard guides you through the initial installation of MySQL products. As the following figure shows, the initial setup is a one-time activity in the overall process. MySQL Installer detects existing MySQL products installed on the host during its initial setup and adds them to the list of products to be managed.

Figure 5.1 MySQL Installer Process Overview



MySQL Installer extracts configuration files (described later) to the hard drive of the host during the initial setup. Although MySQL Installer is a 32-bit application, it can install both 32-bit and 64-bit binaries.

The initial setup adds a link to the Start menu under the **MySQL** group. Click **Start, All Programs, MySQL, MySQL Installer** to open MySQL Installer.

MySQL Installer Licensing and Support Authentication

MySQL Installer requires you to accept the license agreement before it will install new MySQL packages. After you accept the terms of the agreement, you can add, update, reconfigure, and remove all of the products and features provided by the MySQL Installer edition you downloaded.

For the commercial edition, entering your My Oracle Support (MOS) credentials is optional when installing bundled MySQL products, but your credentials are required when choosing unbundled MySQL products that MySQL Installer must download. An unbundled product is any MSI file that you download using MySQL Installer after the initial setup. Your credentials must match the user name and password that you have registered with Oracle for access to the support site.

Choosing a Setup Type

During the initial setup, you are prompted to select the MySQL products to be installed on the host. One alternative is to use a predetermined setup type that matches your setup requirements. Choosing one of the following setup types determines the initial installation only and does not limit your ability to install or update MySQL products for Windows later:

- **Developer Default:** Install the following products that compliment application development with MySQL:
 - [MySQL Server](#) (Installs the version that you selected when you downloaded MySQL Installer.)

- [MySQL Shell](#)
- [MySQL Router](#)
- [MySQL Workbench](#)
- [MySQL for Visual Studio](#)
- [MySQL for Excel](#)
- [MySQL Notifier](#)
- [MySQL Connectors](#) (.NET / Python / ODBC / Java / C / C++)
- [MySQL Utilities](#)
- [MySQL Documentation](#)
- [MySQL Samples and Examples](#)
- **Server only:** Only install the MySQL server. This setup type installs the general availability (GA) or development release server that you selected when you downloaded MySQL Installer. It uses the default installation and data paths.
- **Client only:** Only install the most recent MySQL applications and MySQL connectors. This setup type is similar to the [Developer Default](#) type, except that it does not include MySQL server or the client programs typically bundled with the server, such as `mysql` or `mysqladmin`.
- **Full:** Install all available MySQL products.
- **Custom** The custom setup type enables you to filter and select individual MySQL products from the [MySQL Installer catalog](#).

Use the [Custom](#) setup type to install:

- A product or product version that is not available from the usual download locations. The catalog contains all product releases, including the other releases between pre-release (or development) and GA.
- An instance of MySQL server using an alternative installation path, data path, or both. For instructions on how to adjust the paths, see [Section 5.3.2.3, "Setting Alternative Server Paths"](#).
- Two or more MySQL server versions on the same host at the same time (for example, 5.6, 5.7, and 8.0).
- A specific combination of products and features not offered as a predetermine setup type. For example, you can install a single product, such as MySQL Workbench, instead of installing all client applications for Windows.

Path Conflicts

When the default installation or data folder (required by MySQL server) for a product to be installed already exists on the host, the wizard displays the **Path Conflict** step to identify each conflict and enable you to take action to avoid having files in the existing folder overwritten by the new installation. You see this step in the initial setup only when MySQL Installer detects a conflict.

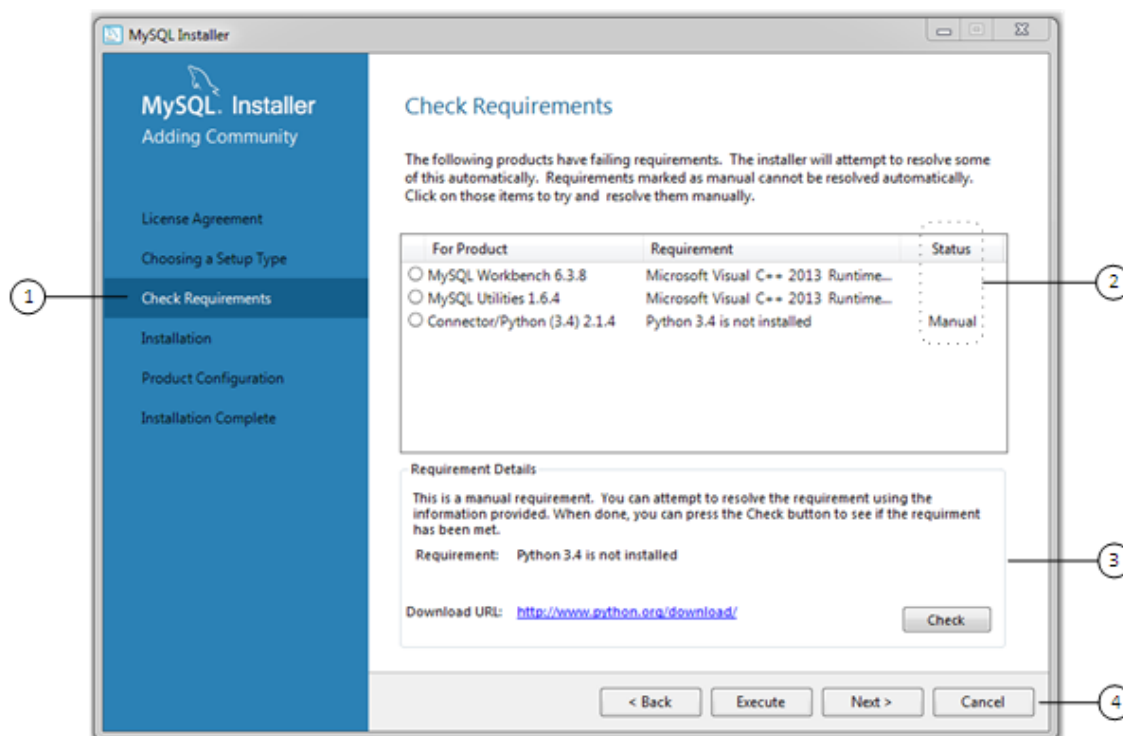
To resolve the path conflict, do one of the following:

- Select a product from the list to display the conflict options. A warning symbol indicates which path is in conflict. Use the browse button to choose a new path and then click **Next**.
- Click **Back** to choose a different setup type or product version, if applicable. The **Custom** setup type enables you to select individual product versions.
- Click **Next** to ignore the conflict and overwrite files in the existing folder.
- Delete the existing product. Click **Cancel** to stop the initial setup and close MySQL Installer. Open MySQL Installer again from the Start menu and delete the installed product from the host using the Delete operation from the dashboard.

Check Requirements

MySQL Installer uses entries in the `package-rules.xml` file to determine whether the prerequisite software for each product is installed on the host. When the requirements check fails, MySQL Installer displays the **Check Requirements** step to help you update the host. The following figure identifies and describes the key areas of this step.

Figure 5.2 Check Requirements



Description of Check Requirements Elements

1. Shows the current step in the initial setup. Steps in this list may change slightly depending on the products already installed on the host, the availability of prerequisite software, and the products to be installed on the host.
2. Lists all pending installation requirements by product and indicates the status as follows:
 - A blank space in the **Status** column means that MySQL Installer can attempt to download and install the required software for you.

- The word *Manual* in the **Status** column means that you must satisfy the requirement manually. Select each product in the list to see its requirement details.
3. Describes the requirement in detail to assist you with each manual resolution. When possible, a download URL is provided. After you download and install the required software, click **Check** to verify that the requirement has been met.
 4. Provides the following set operations to proceed:
 - **Back** – Return to the previous step. This action enables you to select a different the setup type.
 - **Execute** – Have MySQL Installer attempt to download and install the required software for all items without a manual status. Manual requirements are resolved by you and verified with the **Check** button.
 - **Next** – Do not execute the request to apply the requirements automatically and proceed to the installation without including the products that fail the check requirements step.
 - **Cancel** – Stop the installation of MySQL products. Because MySQL Installer is already installed, the initial setup begins again when you open MySQL Installer from the Start menu and click **Add** from the dashboard. For a description of the available management operations, see [Product Catalog](#).

MySQL Installer Configuration Files

All MySQL Installer files are located within the `C:\Program Files (x86)` and `C:\ProgramData` folders. The following table describes the files and folders that define MySQL Installer as a standalone application.

Note

Installed MySQL products are neither altered nor removed when you update or uninstall MySQL Installer.

Table 5.2 MySQL Installer Configuration Files

File or Folder	Description	Folder Hierarchy
MySQL Installer for Windows	This folder contains all of the files needed to run MySQL Installer and <code>MySQLInstallerConsole.exe</code> , a command-line program with similar functionality.	<code>C:\Program Files (x86)</code>
Templates	The <code>Templates</code> folder has one file for each version of MySQL server. Template files contain keys and formulas to calculate some values dynamically.	<code>C:\ProgramData\MySQL\MySQL Installer for Windows\Manifest</code>
<code>package-rules.xml</code>	This file contains the prerequisites for every product to be installed.	<code>C:\ProgramData\MySQL\MySQL Installer for Windows\Manifest</code>
<code>products.xml</code>	The <code>products</code> file (or product catalog) contains a list of all products available for download.	<code>C:\ProgramData\MySQL\MySQL Installer for Windows\Manifest</code>
Product Cache	The <code>Product Cache</code> folder contains all standalone MSI files bundled with the full package or downloaded afterward.	<code>C:\ProgramData\MySQL\MySQL Installer for Windows</code>

5.3.2 Installation Workflow

MySQL Installer provides a wizard-like tool to install and configure new MySQL products for Windows. Unlike the initial setup, which runs only once, MySQL Installer invokes the wizard each time you download or install a new product. For first-time installations, the steps of the initial setup proceed directly into the steps of the installation.

Note

Full permissions are granted to the user executing MySQL Installer to all generated files, such as `my.ini`. This does not apply to files and directories for specific products, such as the MySQL server data directory in `%ProgramData%` that is owned by `SYSTEM`.

Products installed and configured on a host follow a general pattern that might require your input during the various steps. MySQL Installer loads all selected products together using the following workflow:

- Product download. If you installed the full (not web) MySQL Installer package, all MSI files were loaded to the `Product Cache` folder during the initial setup and are not downloaded again. Otherwise, the status of each product changes from `Downloading` to `Downloaded`.
- Product installation. The status of each product in the list changes from `Ready to Install` to `Installing` to `Complete`. During the process, click **Show Details** to view the installation actions.

If you cancel the installation at this point, the products are installed, but the server (if installed) is not yet configured. To restart the server configuration, open MySQL Installer from the Start menu and click the **Reconfigure** link next to the appropriate server in the dashboard.

- Product configuration. This step applies to MySQL server and samples in most cases. The status for each item in the list should indicate, `Ready to Configure`. Click **Next** to begin the step-by-step configuration of all items in the list.

The configuration options presented during this step depend on which version of the database you selected to install. After the installation completes, you can reconfigure MySQL server from the MySQL Installer dashboard.

- Installation complete. This step finalizes the installation and enables you to start some applications when the installation finishes.

5.3.2.1 InnoDB Cluster Sandbox Test Setup

You have two options to implement a high-availability solution when you install MySQL Server 5.7.17 or higher using MySQL Installer:

- Standalone MySQL Server / Classic MySQL Replication (default)

Select this option to begin the initial configuration of a standalone MySQL server. You can configure multiple servers with classic MySQL Replication manually or use MySQL Shell 1.0.9 or higher to configure a production InnoDB cluster.

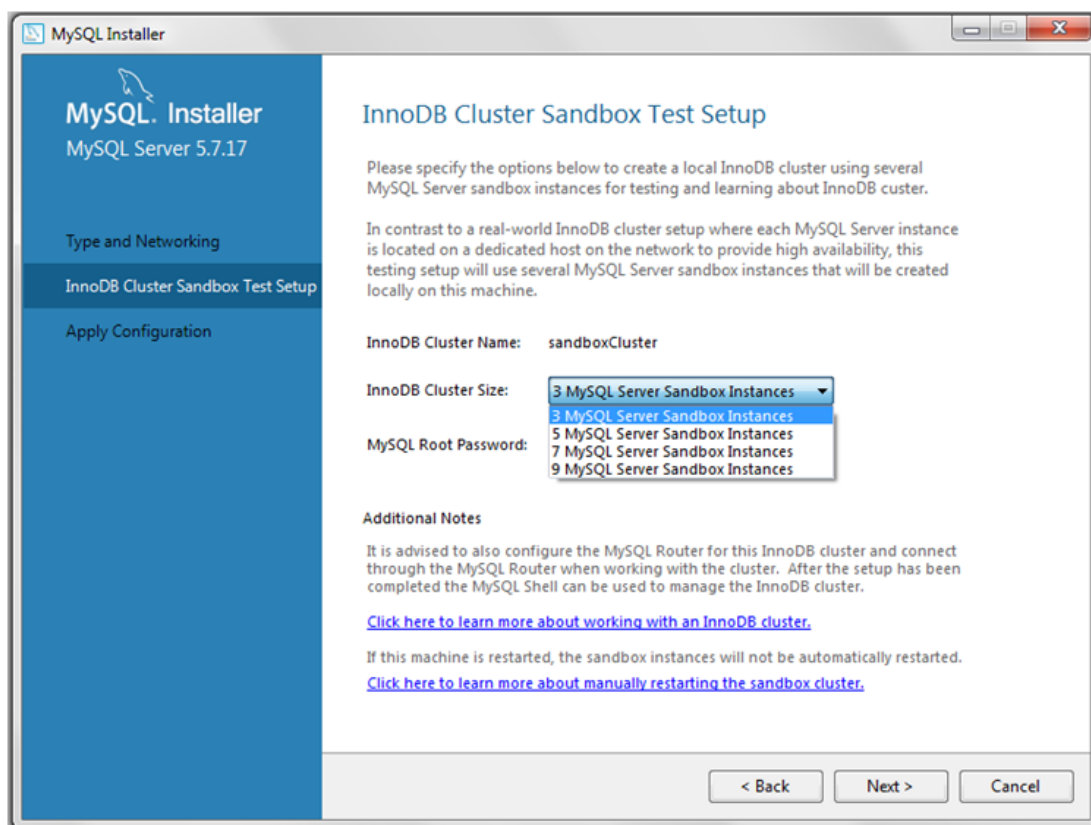
For a description of the server configuration options that apply to a standalone MySQL server on Windows, see [Section 5.3.2.2, “Server Configuration”](#).

- InnoDB Cluster Sandbox Test Setup (for testing only)

Select this option to create and configure InnoDB cluster sandbox instances locally for testing. You can configure an InnoDB cluster sandbox to have three, five, seven, or nine MySQL server instances. Use

the **Reconfigure** quick action in the MySQL Installer toolbar to adjust the number of instances in the InnoDB cluster after the configuration has finished.

Figure 5.3 InnoDB Cluster Sandbox Test Setup



The InnoDB cluster sandbox, named `sandboxCluster` by default, is available on selected ports. After the configuration executes, click the **Summary** tab to view the specific ports that apply to your cluster.

You can use MySQL Installer to install MySQL Shell 1.0.9, if it is not installed. MySQL Shell enables you to manage the sandbox instances. To connect with the MySQL Shell on port 3310, execute the following command:

```
shell> mysqlsh root@localhost:3310
```

MySQL Installer also provides a wizard for configuring MySQL Router to connect to the test InnoDB cluster that was created in this step. For configuration details, see [MySQL Router Configuration](#). To learn more about MySQL Router operations, see [Routing for MySQL InnoDB cluster](#).

5.3.2.2 Server Configuration

MySQL Installer handles the initial configuration of the MySQL server. For example:

- It creates the configuration file (`my.ini`) that is used to configure the MySQL server. The values written to this file are influenced by choices you make during the installation process.

Note

Some definitions are host dependent. For example, `query_cache` is enabled if the host has fewer than three cores.

- By default, a Windows service for the MySQL server is added.
- Provides default installation and data paths for MySQL server. For instructions on how to change the default paths, see [Section 5.3.2.3, “Setting Alternative Server Paths”](#).
- It can optionally create MySQL server user accounts with configurable permissions based on general roles, such as DB Administrator, DB Designer, and Backup Admin. It optionally creates a Windows user named `Mysq1Sys` with limited privileges, which would then run the MySQL Server.

User accounts may also be added and configured in MySQL Workbench.

- Checking **Show Advanced Options** allows additional **Logging Options** to be set. This includes defining custom file paths for the error log, general log, slow query log (including the configuration of seconds it requires to execute a query), and the binary log.

During the configuration process, click **Next** to proceed to the next step or **Back** to return to the previous step. Click **Execute** at the final step to apply the server configuration.

The sections that follow describe the server configuration options that apply to MySQL server on Windows. The server version you installed will determine which steps and options you can configure.

- [Type and Networking](#)
- [Accounts and Roles](#)
- [Windows Service](#)
- [Plugins and Extensions](#)
- [Advanced Options](#)
- [Apply Server Configuration](#)

Type and Networking

- Server Configuration Type


Choose the MySQL server configuration type that describes your setup. This setting defines the amount of system resources (memory) that will be assigned to your MySQL server instance.

- **Development:** A machine that will host many other applications, and typically this is your personal workstation. This option configures MySQL to use the least amount of memory.
- **Server:** Several other applications will be running on this machine, such as a web server. This option configures MySQL to use a medium amount of memory.
- **Dedicated:** A machine that is dedicated to running the MySQL server. Because no other major applications will run on this server, such as a web server, this option configures MySQL to use the majority of available memory.

- Connectivity

Connectivity options control how the connection to MySQL is made. Options include:

- **TCP/IP:** You may enable TCP/IP Networking here as otherwise only local host connections are allowed. Also define the **Port Number** and whether to open the firewall port for network access. If the

port number is in use already, you will see the information icon () next to the default value and **Next** is disabled until you provide a new port number.

- **Named Pipe:** Enable and define the pipe name, similar to using the `--enable-named-pipe` option.
- **Shared Memory:** Enable and then define the memory name, similar to using the `--shared-memory` option.
- Advanced Configuration

Check **Show Advanced Options** to set additional logging options in a later step. This includes defining custom file paths for the error log, general log, slow query log (including the configuration of seconds it requires to execute a query), and the binary log.


- MySQL Enterprise Firewall (Commercial Edition only)

The **Enable Enterprise Firewall** check box is selected by default. For post-installation instructions, see [MySQL Enterprise Firewall](#).

Accounts and Roles

- Root Account Password

Assigning a root password is required and you will be asked for it when performing other MySQL Installer operations. Password strength is evaluated when you repeat the password in the box provided. For descriptive information regarding password requirements or status, move your mouse pointer over

the information icon () when it appears.

- MySQL User Accounts

Optionally, you can create additional MySQL user accounts with predefined user roles. Each predefined role, such as DB Admin, are configured with their own set of privileges. For example, the DB Admin role has more privileges than the DB Designer role. Click the **Role** drop-down list for a description of each role.

Note

If the MySQL server is installed, then you must also enter the current root password.

Windows Service

On the Windows platform, MySQL server can run as a named service managed by the operating system and be configured to start up automatically when Windows starts. Alternatively, you can configure MySQL server to run as an executable program that requires manual configuration.

- **Configure MySQL server as a Windows service** (Selected by default.)

When the default configuration option is selected, you can also select the following:

- **Start the MySQL Server at System Startup**

When selected (default), the service startup type is set to Automatic; otherwise, the startup type is set to Manual.

- **Run Windows Service as**

When **Standard System Account** is selected (default), the service logs on as Network Service.

The **Custom User** option must have privileges to log on to Microsoft Windows as a service. The **Next** button will be disabled until this user is configured with the required privileges.

A custom user is configured in Windows by searching for "local security policy" in the Start menu. In the Local Security Policy window, select **Local Policies, User Rights Assignment**, and then **Log On As A Service** to open the property dialog. Click **Add User or Group** to add the custom user and then click **OK** in each dialog to save the changes.

- Deselect the Windows Service option

Plugins and Extensions

The **Plugins and Extensions** step is visible during a new installation of MySQL server version 5.7.12 (or higher) only. If you are upgrading from a previous MySQL version, then you need to open MySQL Installer again and select the **Reconfigure** MySQL server option.

- **Enable X Protocol / MySQL as a Document Store** (Selected by default.)

When the X Protocol option is selected, MySQL Installer loads and starts the X Plugin. Without the X Plugin running, X Protocol clients cannot connect to the server.

- **Port Number:** [33060](#)

Requires an unused port. The default port number is 33060.

- **Open Firewall port for network access**

Open by default when the X Protocol is selected.

For more information about using MySQL as a document store and the X Plugin, see [Key Concepts](#) and [X Plugin](#).

Advanced Options

This step is available if the **Show Advanced Configuration** check box was selected during the **Type and Networking** step. To enable this step now, click **Back** to return to the **Type and Networking** step and select the check box.

Advanced configuration options are related to the following MySQL log files:

- [Error log](#)
- [General log](#)
- [Slow query log](#)
- [Bin log](#)

Apply Server Configuration

All configuration settings are applied to the MySQL server when you click **Execute**. Use the **Configuration Steps** tab to follow the progress of each action; the icon for each toggles from white to green on success. Otherwise, the process stops and displays an error message if an individual action times out. Click the **Log** tab to view the log.

When the installation is done and you click **Finish**, MySQL Installer and the installed MySQL products are added to the Microsoft Windows Start menu under the [MySQL](#) group. Opening MySQL Installer loads the [dashboard](#) where installed MySQL products are listed and other MySQL Installer operations are available.

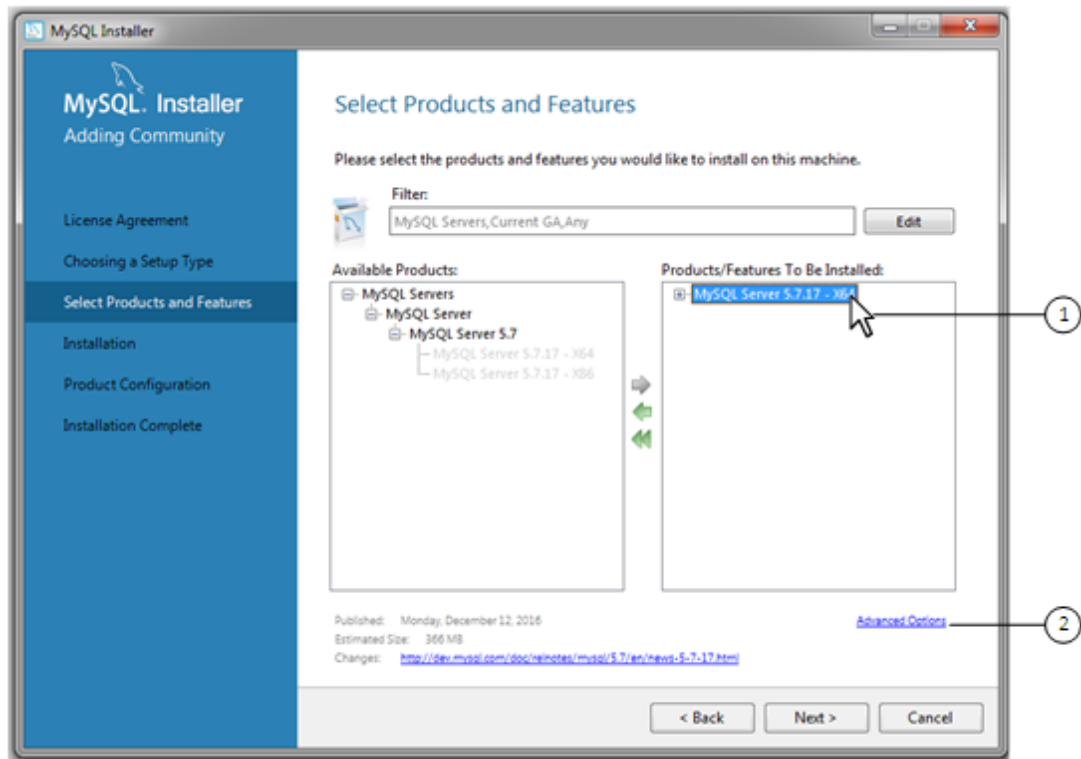
5.3.2.3 Setting Alternative Server Paths

You can change the default installation path, the data path, or both when you install MySQL server. After you have installed the server, the paths cannot be altered without removing and reinstalling the server instance.

To change paths for MySQL server

1. Identify the MySQL server to change and display the **Advanced Options** link.
 - a. Navigate to the **Select Products and Features** step by doing one of the following:
 - i. If this is an [initial setup](#), select the [Custom](#) setup type and click **Next**.
 - ii. If MySQL Installer is installed already, launch it from the Start menu and then click **Add** from the dashboard.
 - b. Click **Edit** to filter the list of products, locate the server instance to be installed in the **Available Products** list.
 - c. With the server instance selected, use the arrow to move the selected server to the **Products/Features To Be Installed** list.
 - d. Click the server to select it. When you select the server, the **Advanced Options** link appears. For details, see the figure that follows.
2. Click **Advanced Options** to open a dialog window with the path-setting options. After setting the path, click **Next** to continue with the configuration steps.

Figure 5.4 Change MySQL Server Path



5.3.2.4 MySQL Applications, Connectors, and Documentation

MySQL Installer provides you with a suite of tools for developing and managing business-critical applications on Windows. The suite consist of applications, connectors, documentation, and samples.

During the [initial setup](#), choose any predetermined setup type, except [Server only](#), to install the latest GA version of the tools. Use the [Custom](#) setup type to install an individual tool or specific version. If MySQL Installer is installed on the host already, use the **Add** operation to select and install tools from the MySQL Installer dashboard.

MySQL Router Configuration

MySQL Installer provides a configuration wizard that can bootstrap an installed instance of MySQL Router 2.1.3 or later to route traffic between MySQL applications and an InnoDB cluster. When configured, MySQL Router runs as a local Windows service. For detailed information about using MySQL Router with an InnoDB cluster, see [Routing for MySQL InnoDB cluster](#).

To configure MySQL Router, do the following:

1. Set up InnoDB cluster. For instructions on how to configure an InnoDB cluster sandbox on the local host using MySQL Installer, see [Section 5.3.2.1, “InnoDB Cluster Sandbox Test Setup”](#). InnoDB cluster requires MySQL Server 5.7.17 or higher.

For general InnoDB cluster information, see [InnoDB Cluster](#).

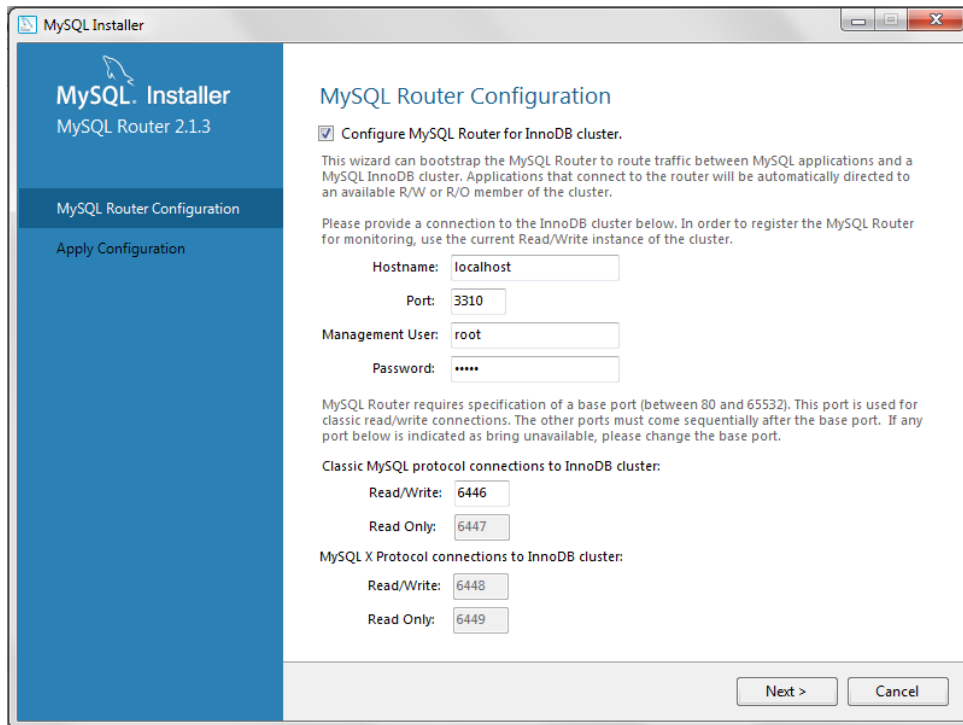
2. Using MySQL Installer, download and install the MySQL Router application. After the installation finishes, the configuration wizard prompts you for information. Select the **Configure MySQL Router for InnoDB cluster** check box to begin the configuration and provide the following configuration values:

- **Hostname:** localhost by default.
- **Port:** The port number of the primary server in the InnoDB cluster. The default is 3310.
- **Management User:** An administrative user with root-level privileges.
- **Password:** The password for the management user.
- **Classic MySQL protocol connections to InnoDB cluster**

Read/Write: Set the first base port number to one that is unused (between 80 and 65532) and the wizard will select the remaining ports for you.

The figure that follows shows an example of the MySQL Router configuration screen.

Figure 5.5 MySQL Router Configuration



3. Click **Next** and then **Execute** to apply the configuration. Click **Finish** to close MySQL Installer or return to the MySQL Installer dashboard.

5.3.3 MySQL Installer Product Catalog and Dashboard

- [Product Catalog](#)
- [MySQL Installer Dashboard](#)
- [Locating Products to Install](#)

This section describes the MySQL Installer product catalog and the dashboard.

Product Catalog

The product catalog stores the complete list of released MySQL products for Microsoft Windows that are available to download from [MySQL](#). By default, and when an Internet connection is present, MySQL Installer updates the catalog daily. You can also update the catalog manually from the dashboard (described later).

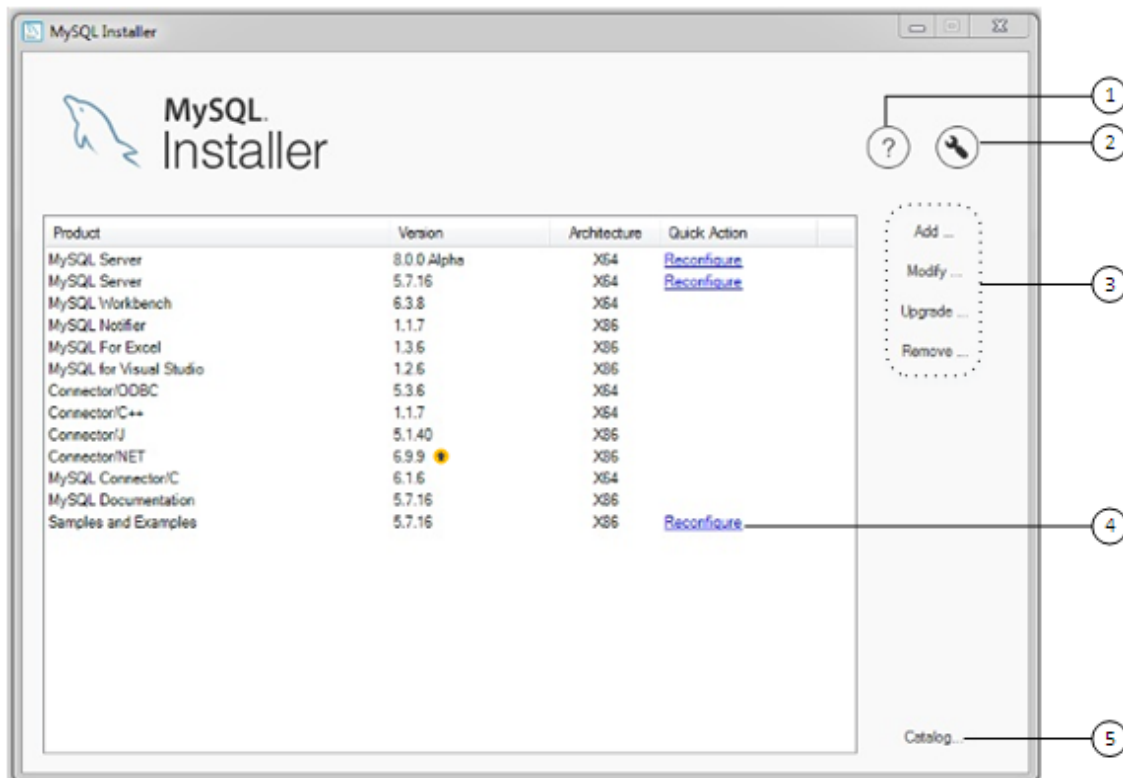
An up-to-date catalog performs the following actions:

- Populates the **Available Products** pane of the Select Products and Features step. This step appears when you select:
 - The **Custom** setup type during the **initial setup**.
 - The **Add** operation from the dashboard.
- Identifies when product updates are available for the installed products listed in the dashboard.

The catalog includes all development releases (Pre-Release), general releases (Current GA), and minor releases (Other Releases). Products in the catalog will vary somewhat, depending on the MySQL Installer edition that you download.

MySQL Installer Dashboard



Figure 5.6 MySQL Installer Dashboard Elements



Description of MySQL Installer Dashboard Elements

1. The About MySQL button (?) shows the current version of MySQL Installer and general information about MySQL. The version number is located above the **Back** button.

Always include this version number when reporting a problem with MySQL Installer.

2. The MySQL Installer Options button () enables you to schedule daily automatic catalog updates. By default, catalog updates are scheduled at the hour when MySQL Installer was first installed. When new products or product versions are available, MySQL Installer adds them to the catalog and then displays an arrow icon () next to the version number of installed products listed in the dashboard.

Use this option to enable or disable automatic catalog updates and to reset the time of day when the MySQL Installer updates the catalog automatically. For specific settings, see the task named [ManifestUpdate](#) in the Windows Task Scheduler.

3. MySQL Installer dashboard operations provide a variety of actions that apply to installed products or products listed in the catalog. To initiate the following operations, first click the operation link and then select the product or products to manage:
 - **Add:** This operation opens the Select Products and Features screen. From there, you can filter the product in the product catalog, select one or more products to download (as needed), and begin the installation. For hints about using the filter, see [Locating Products to Install](#).
 - **Modify:** Use this operation to add or remove the features associated with installed products. Features that you can modify vary in complexity by product. When the **Program Shortcut** check box is selected, the product appears in the Start menu under the [MySQL](#) group.
 - **Upgrade:** This operation loads the Select Products to Upgrade screen and populates it with all the upgrade candidates. An installed product can have more than one upgrade version and requires a current product catalog.

To choose a new product version:

- a. Confirm that the check box next to product name in the **Upgradeable Products** pane has a check mark. Deselect the products that you do not intend to upgrade at this time.
- b. Click a product in the list to highlight it. This action populates the **Upgradeable Versions** pane with the details of each available version for the selected product: version number, published date, and a [Changes](#) link to open the release notes for that version.

MySQL Installer upgrades all of the selected products in one action. Click **Show Details** to view the actions performed by MySQL Installer.

- **Remove** This operation opens the Remove Products screen and populates it with the MySQL products installed on the host. Select the MySQL products you want to remove (uninstall) and then click **Execute** to begin the removal process.

To select products to remove, do one of the following:

- Select the check box for one or more products.
- Select the **Product** check box to select all products.

4. The **Reconfigure** link in the Quick Action column next to each installed server loads the current configuration values for the server and then cycles through all configuration steps enabling you to change the options and values. On completion, MySQL Installer stops the server, applies the configuration changes, and restarts the server for you. For a description of each configuration option, see [Section 5.3.2.2, "Server Configuration"](#).

Installed [Samples and Examples](#) associated with a specific MySQL server version can be also be reconfigured to apply feature-configuration changes, if any. You must provide credentials with root privileges to reconfigure these items.

5. The **Catalog** link enables you to download the latest catalog of MySQL products manually and then to integrate those product changes with MySQL Installer. The catalog-download action does not perform an upgrade of the products already installed on the host. Instead, it returns to the dashboard and displays an arrow icon in the Version column for each installed product that has a newer version. Use the **Upgrade** operation to install the newer product version.

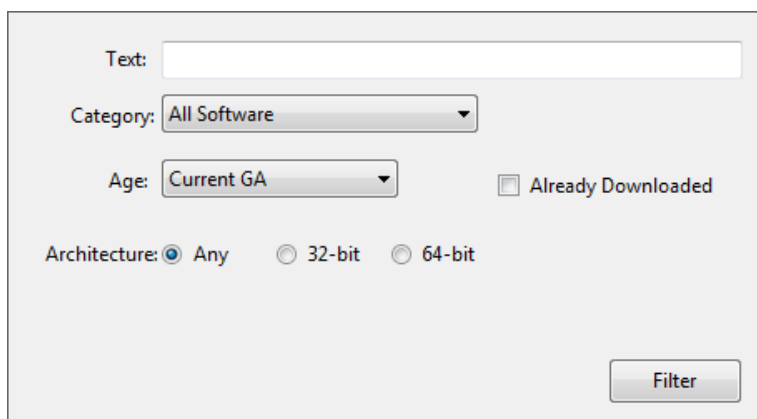
You can also use the **Catalog** link to display the current change history of each product without downloading the new catalog. Select the **Do not update at this time** check box to view the change history only.

Locating Products to Install

MySQL products in the catalog are listed by category: MySQL Servers, Applications, MySQL Connectors, and Documentation. Only the latest GA versions appear in the **Available Products** pane by default. If you are looking for a pre-GA or older version of a product, it may not be visible in the default list.

To change the filter that displays available products in the Select Products and Features screen, click **Edit**.

Figure 5.7 Filter Available Products



The screenshot shows a dialog box for filtering available products. It includes a text input field for searching, a dropdown menu for 'Category' (currently set to 'All Software'), a dropdown menu for 'Age' (currently set to 'Current GA'), a checkbox for 'Already Downloaded' (unchecked), and three radio buttons for 'Architecture' (set to 'Any', with '32-bit' and '64-bit' also available). A 'Filter' button is positioned at the bottom right of the dialog.

Reset the following values to filter the list of available products:

- Text: Filter by text.
- Category: All Software (default), MySQL Servers, Applications, MySQL Connectors, or Documentation (for samples and documentation).
- Age: Pre-Release, Current GA (default), or Other Releases.
- Already Downloaded (the check box is deselected by default).
- Architecture: Any (default), 32-bit, or 64-bit.

5.3.4 MySQLInstallerConsole Reference

[MySQLInstallerConsole.exe](#) provides command-line functionality that is similar to MySQL Installer. It is installed when MySQL Installer is initially executed and then available within the [MySQL Installer](#)

directory. Typically, that is in `C:\Program Files (x86)\MySQL\MySQL Installer\`, and the console must be executed with administrative privileges.

To use, invoke the command prompt with administrative privileges by choosing **Start, Accessories**, then right-click on **Command Prompt** and choose **Run as administrator**. And from the command line, optionally change the directory to where `MySQLInstallerConsole.exe` is located:

```
C:\> cd Program Files (x86)\MySQL\MySQL Installer for Windows
C:\Program Files (x86)\MySQL\MySQL Installer for Windows> MySQLInstallerConsole.exe help
===== Start Initialization =====
MySQL Installer is running in Community mode

Attempting to update manifest.
Initializing product requirements
Loading product catalog
Checking for product catalog snippets
Checking for product packages in the bundle
Categorizing product catalog
Finding all installed packages.
Your product catalog was last updated at 11/1/2016 4:10:38 PM
===== End Initialization =====

The following commands are available:

Configure - Configures one or more of your installed programs.
Help      - Provides list of available commands.
Install   - Install and configure one or more available MySQL programs.
List      - Provides an interactive way to list all products available.
Modify    - Modifies the features of installed products.
Remove    - Removes one or more products from your system.
Status    - Shows the status of all installed products.
Update    - Update the current product catalog.
Upgrade   - Upgrades one or more of your installed programs.
```

`MySQLInstallerConsole.exe` supports the following options, which are specified on the command line:

Note

Configuration block values that contain a colon (":") must be wrapped in double quotes. For example, `installdir="C:\MySQL\MySQL Server 8.0"`.

- `configure [product1]:[setting]=[value]; [product2]:[setting]=[value]; [...]`

Configure one or more MySQL products on your system. Multiple `setting=value` pairs can be configured for each product.

Switches include:

- `-showsettings` : Displays the available options for the selected product, by passing in the product name after `-showsettings`.
- `-silent` : Disable confirmation prompts.

```
C:\> MySQLInstallerConsole configure -showsettings server
C:\> MySQLInstallerConsole configure server:port=3307
```

- `help [command]`

Displays a help message with usage examples, and then exits. Pass in an additional command to receive help specific to that command.

```
C:\> MySQLInstallerConsole help
C:\> MySQLInstallerConsole help install
```

- `install [product]:[features]:[config block]:[config block]:[config block]; [...]`

Install one or more MySQL products on your system.

Switches and syntax options include:

- `-type=[SetupType]` : Installs a predefined set of software. The "SetupType" can be one of the following:

Note

Non-custom setup types can only be chosen if no other MySQL products are installed.

- **Developer**: Installs a complete development environment.
- **Server**: Installs a single MySQL server
- **Client**: Installs client programs and libraries
- **Full**: Installs everything
- **Custom**: Installs user selected products. This is the default option.
- `-showsettings` : Displays the available options for the selected product, by passing in the product name after `-showsettings`.
- `-silent` : Disable confirmation prompts.
- `[config block]`: One or more configuration blocks can be specified. Each configuration block is a semicolon separated list of key value pairs. A block can include either a "config" or "user" type key, where "config" is the default type if one is not defined.

Configuration block values that contain a colon (":") must be wrapped in double quotes. For example, `installdir="C:\MySQL\MySQL Server 8.0"`.

Only one "config" type block can be defined per product. A "user" block should be defined for each user that should be created during the product's installation.

Note

Adding users is not supported when a product is being reconfigured.

- `[feature]`: The feature block is a semicolon separated list of features, or '*' to select all features.

```
C:\> MySQLInstallerConsole install server;5.6.25:*:port=3307;serverid=2:type=user;username=foo;password=
C:\> MySQLInstallerConsole install server;5.6.25;x64 -silent
```

An example that passes in additional configuration blocks, broken up by ^ to fit this screen:

```
C:\> MySQLInstallerConsole install server;5.6.25;x64:*:type=config;openfirewall=true; ^
```

```
generallog=true;binlog=true;serverid=3306;enable_tcpip=true;port=3306;rootpasswd=pass; ^
installdir="C:\MySQL\MySQL Server 5.6":type=user;datadir="C:\MySQL\data";username=foo;password=bar
```

- `list`

Lists an interactive console where all of the available MySQL products can be searched. Execute `MySQLInstallerConsole list` to launch the console, and enter in a substring to search.

```
C:\> MySQLInstallerConsole list
```

- `modify [product1:-removelist/+addlist] [product2:-removelist/+addlist] [...]`

Modifies or displays features of a previously installed MySQL product.

- `-silent` : Disable confirmation prompts.

```
C:\> MySQLInstallerConsole modify server
C:\> MySQLInstallerConsole modify server:+documentation
C:\> MySQLInstallerConsole modify server:-debug
```

- `remove [product1] [product2] [...]`

Removes one or more products from your system.

- `*` : Pass in `*` to remove all of the MySQL products.
- `-continue` : Continue the operation even if an error occurs.
- `-silent` : Disable confirmation prompts.

```
C:\> MySQLInstallerConsole remove *
C:\> MySQLInstallerConsole remove server
```

- `status`

Provides a quick overview of the MySQL products that are installed on the system. Information includes product name and version, architecture, date installed, and install location.

```
C:\> MySQLInstallerConsole status
```

- `update`

Downloads the latest MySQL product catalog to your system. On success, the download catalog will be applied the next time either `MySQLInstaller` or `MySQLInstallerConsole` is executed.

```
C:\> MySQLInstallerConsole update
```

Note

The **Automatic Catalog Update** GUI option executes this command from the Windows Task Scheduler.

- `upgrade [product1:version] [product2:version] [...]`

Upgrades one or more products on your system. Syntax options include:

- `*` : Pass in `*` to upgrade all products to the latest version, or pass in specific products.
- `!` : Pass in `!` as a version number to upgrade the MySQL product to its latest version.
- `-silent` : Disable confirmation prompts.

```
C:\> MySQLInstallerConsole upgrade *
C:\> MySQLInstallerConsole upgrade workbench:6.3.5
C:\> MySQLInstallerConsole upgrade workbench:!
C:\> MySQLInstallerConsole upgrade workbench:6.3.5 excel:1.3.2
```

5.4 MySQL Notifier

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 system tray. MySQL Notifier also gives quick access to MySQL Workbench through its context menu.

The MySQL Notifier is installed by MySQL Installer, and (by default) will start-up when Microsoft Windows is started.

To install, download and execute the [MySQL Installer](#), be sure the MySQL Notifier product is selected, then 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's green if all monitored MySQL Server instances are running, or red 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.

5.4.1 MySQL Notifier Usage

MySQL Notifier resides in the system tray and provides visual status information for your MySQL server instances. A green icon is displayed at the top left corner of the tray icon if the current MySQL server is running, or a red icon if the service is stopped.

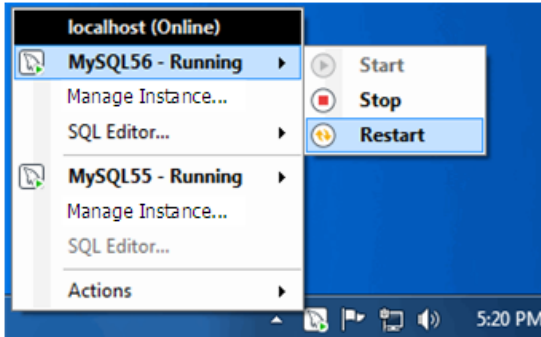
MySQL Notifier automatically adds discovered MySQL services on the local machine, and each service is saved and configurable. By default, the **Automatically add new services whose name contains** option is enabled and set to `mysql`. Related **Notifications Options** include being notified when new services are

either discovered or experience status changes, and are also enabled by default. And uninstalling a service will also remove the service from MySQL Notifier.

Clicking the system tray icon will reveal several options, as the follow figures show:

The Service Instance menu is the main MySQL Notifier window, and enables you to Stop, Start, and Restart the MySQL server.

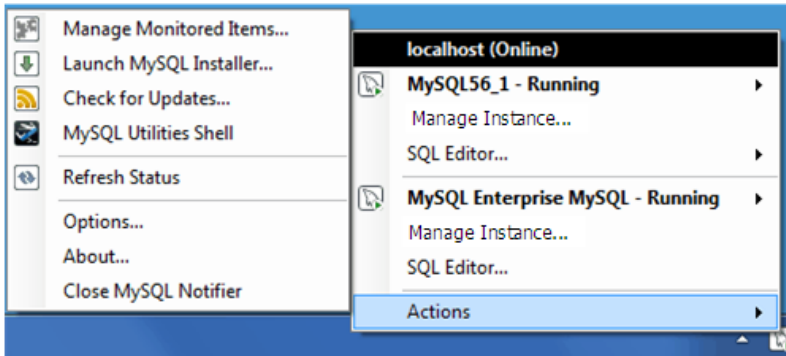
Figure 5.8 MySQL Notifier Service Instance menu



The **Actions** menu includes several links to external applications (if they are installed), and a **Refresh Status** option to manually refresh the status of all monitored services (in both local and remote computers) and MySQL instances.

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

Figure 5.9 MySQL Notifier Actions menu

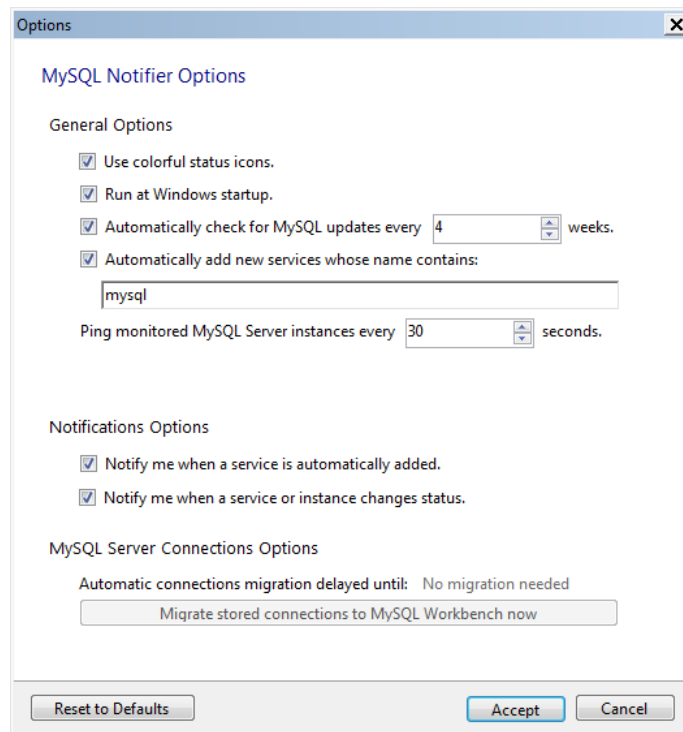


The **Actions, Options** menu configures MySQL Notifier and includes options to:

- **Use colorful status icons:** Enables a colorful style of icons for the tray of MySQL Notifier.
- **Run at Windows Startup:** Allows the application to be loaded when Microsoft Windows starts.
- **Automatically Check For Updates Every # Weeks:** Checks for a new version of MySQL Notifier, and runs this check every # 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.

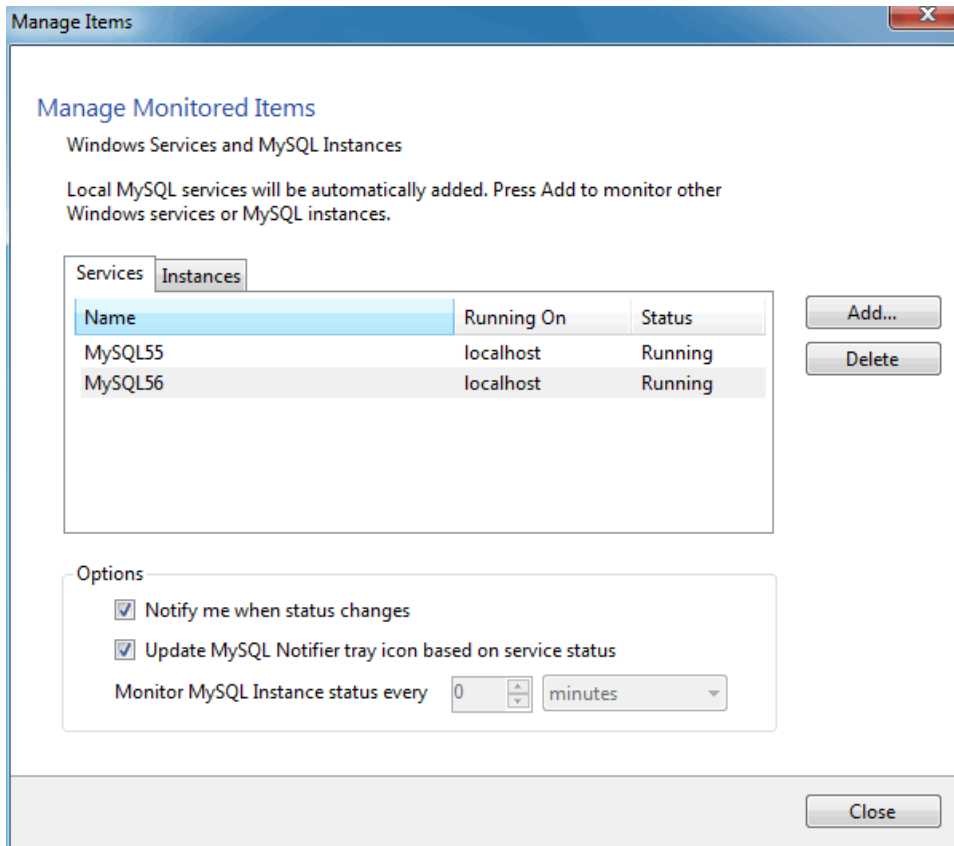
- **Ping monitored MySQL Server instances every # seconds:** The interval (in seconds) to ping monitored MySQL Server instances for status changes. Longer intervals might be necessary if the list of monitored remote instances is large.
- **Notify me when a service is automatically added:** Will display a balloon notification from the taskbar when a newly discovered service is added to the monitored services list.
- **Notify me when a service changes status:** Will display a balloon notification from the taskbar when a monitored service changes its status.
- **Automatic connections migration delayed until:** When there are connections to migrate, postpone the migration by one hour, one day, one week, one month, or indefinitely.

Figure 5.10 MySQL Notifier Options menu

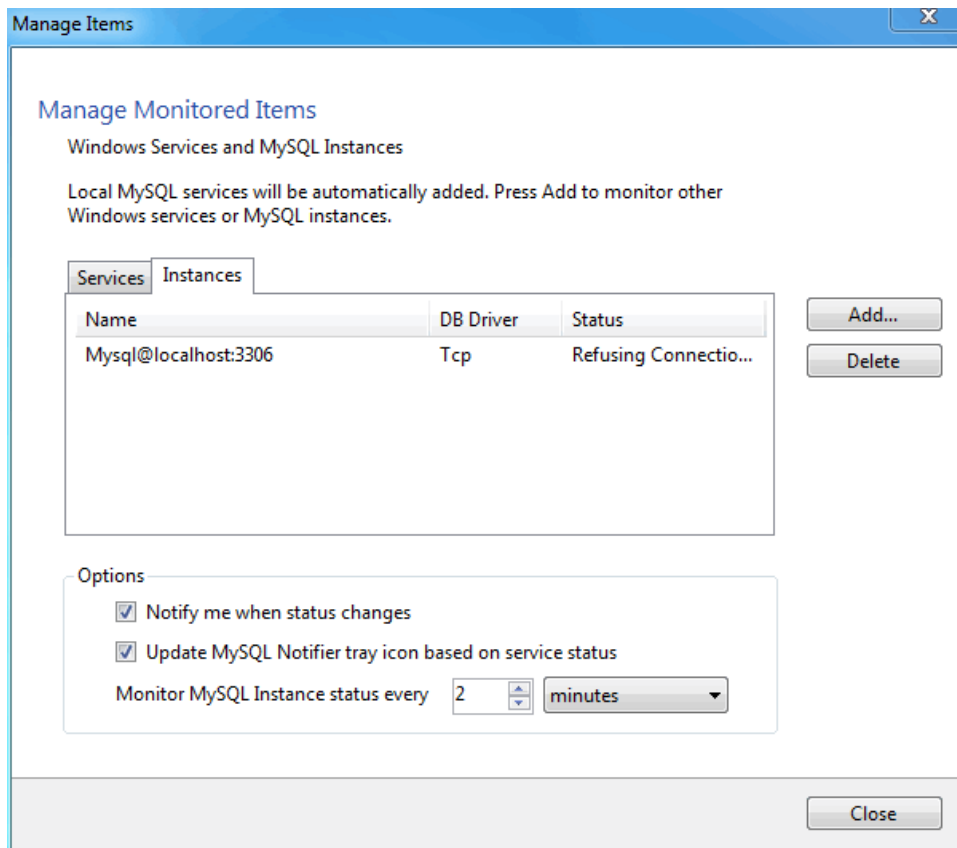


The **Actions, Manage Monitored Items** menu enables you to configure the monitored services and MySQL instances. First, with the **Services** tab open:

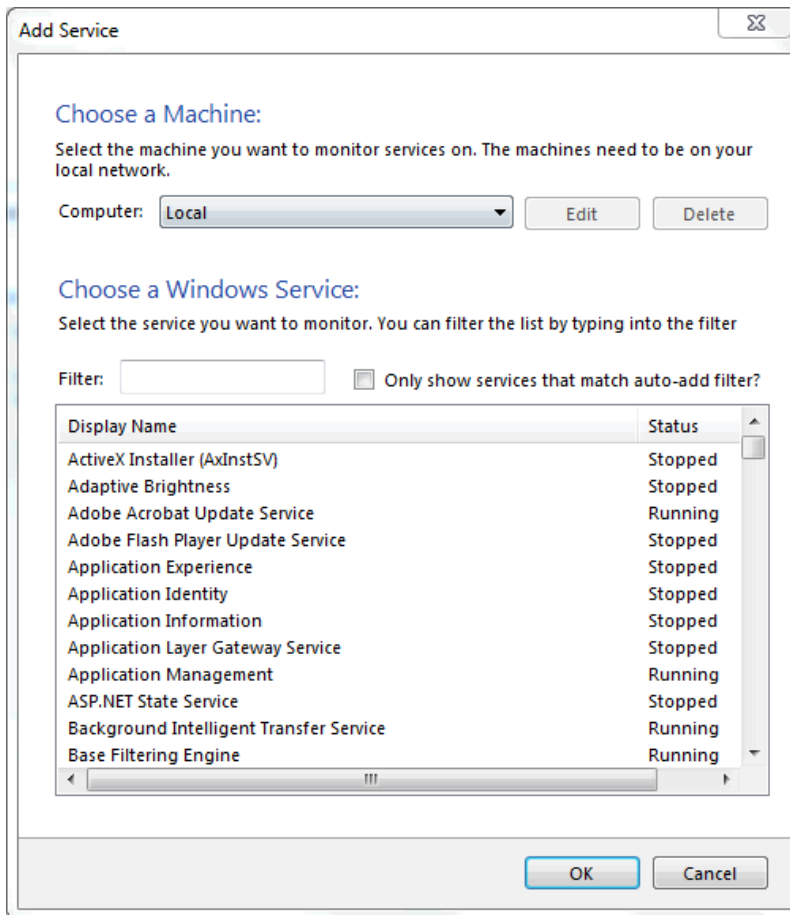
Figure 5.11 MySQL Notifier Manage Services menu



The **Instances** tab is similar:

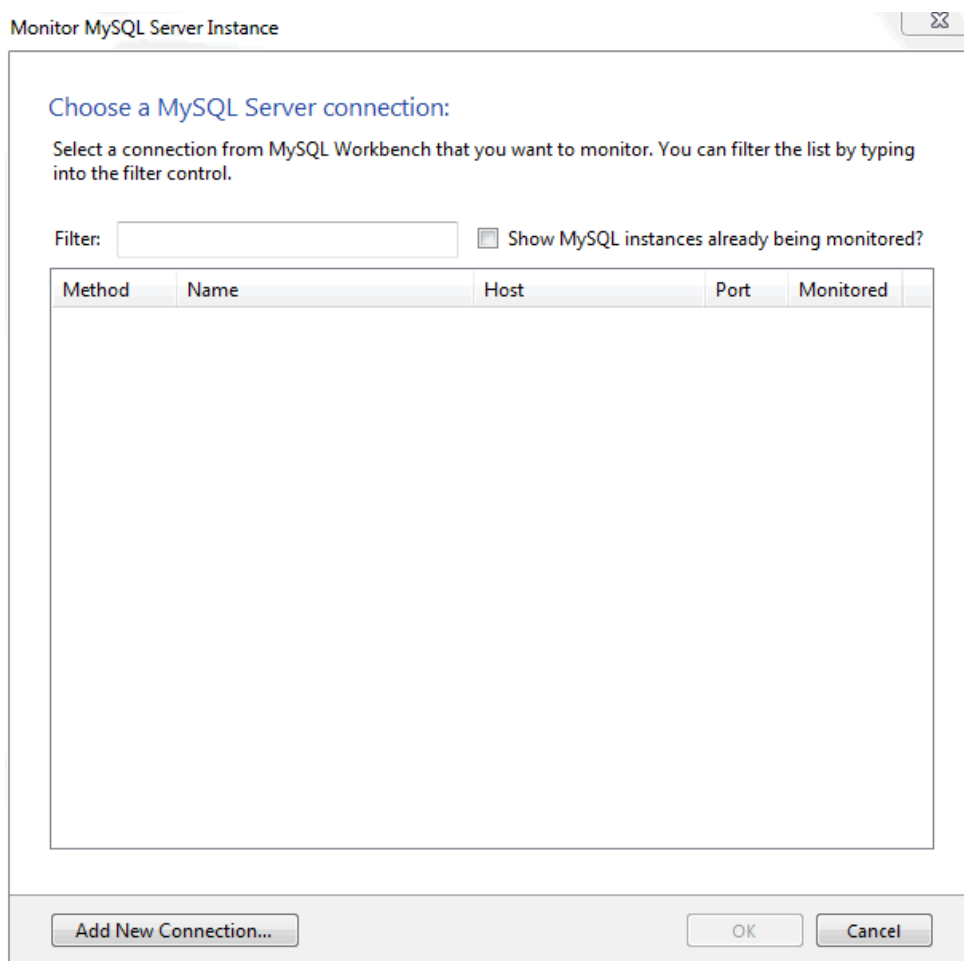
Figure 5.12 MySQL Notifier Manage Instances menu

Adding a service or instance (after clicking **Add** in the **Manage Monitored Items** window) enables you to select a running Microsoft Windows service or instance connection, and configure MySQL Notifier to monitor it. Add a new service or instance by clicking service name from the list, then **OK** to accept. Multiple services and instances may be selected.

Figure 5.13 MySQL Notifier Adding new services

Add instances:

Figure 5.14 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\Running\Oracle\MySQL Notifier\settings.config" where *YourUsername* is your system's 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 5.4.2, “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.

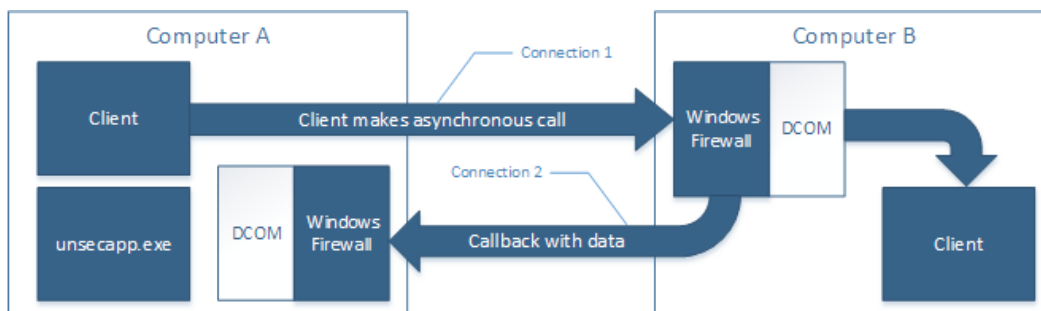
5.4.2 Setting Up Remote Monitoring in MySQL Notifier

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.

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.

Figure 5.15 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's left panel, 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]\MySQLNotifier.exe
```

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**.

- 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.

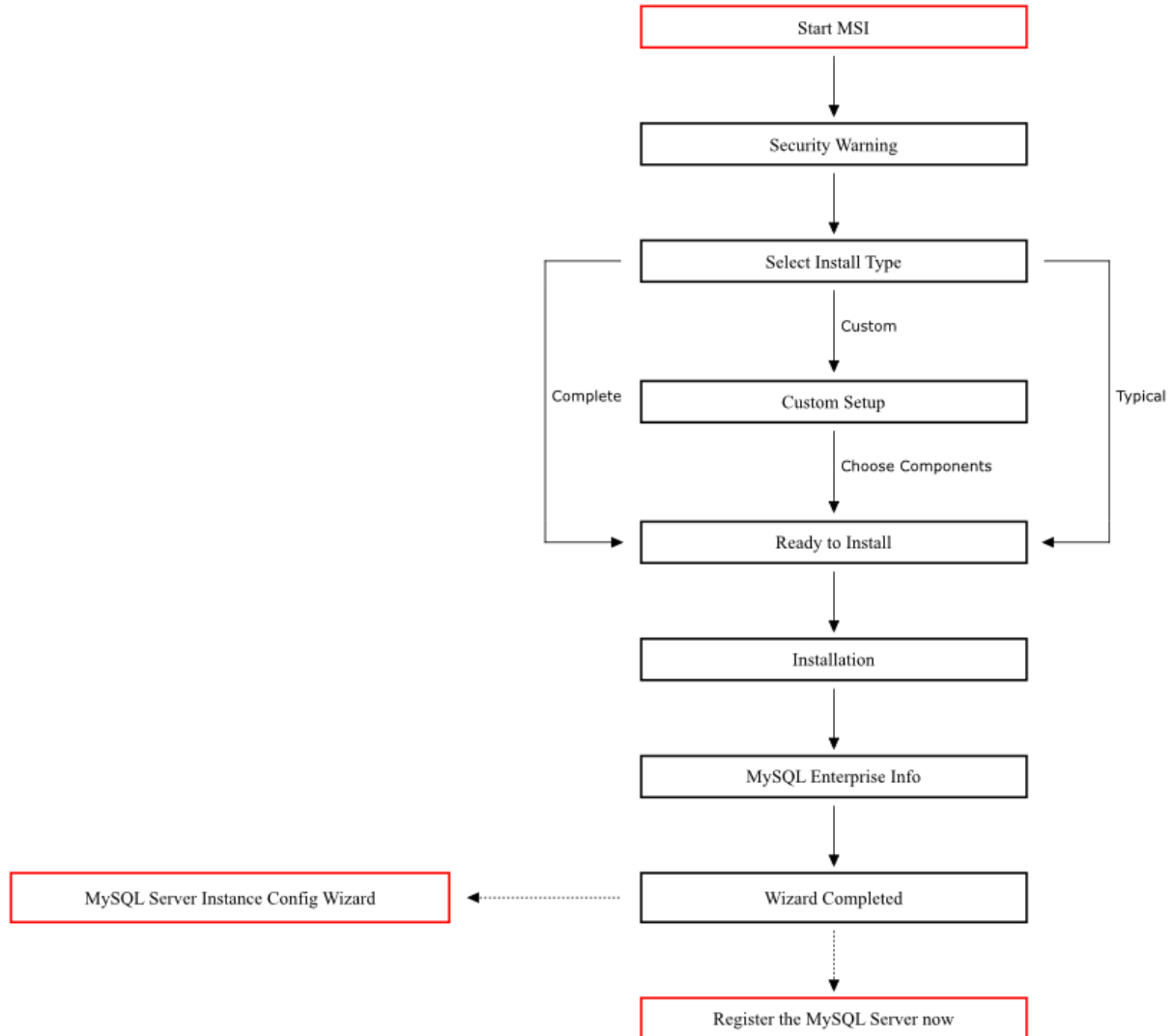
5.5 Installing MySQL on Microsoft Windows Using an MSI Package

The MSI package is designed to install and configure MySQL in such a way that you can immediately get started using MySQL.

The MySQL Installation Wizard and MySQL Configuration Wizard are available in the Complete install package, which is recommended for most standard MySQL installations. Exceptions include users who need to install multiple instances of MySQL on a single server host and advanced users who want complete control of server configuration.

- For information on installing using the GUI MSI installer process, see [Section 5.5.1, “Using the MySQL Installation Wizard”](#).
- For information on installing using the command line using the MSI package, see [Section 5.5.2, “Automating MySQL Installation on Microsoft Windows Using the MSI Package”](#).
- If you have previously installed MySQL using the MSI package and want to remove MySQL, see [Section 5.5.3, “Removing MySQL When Installed from the MSI Package”](#).

The workflow sequence for using the installer is shown in the figure below:

Figure 5.16 Installation Workflow for Windows Using MSI Installer**Note**

Microsoft Windows XP and later include a firewall which specifically blocks ports. If you plan on using MySQL through a network port then you should open and create an exception for this port before performing the installation. To check and if necessary add an exception to the firewall settings:

1. First ensure that you are logged in as an Administrator or a user with Administrator privileges.
2. Go to the **Control Panel**, and double click the **Windows Firewall** icon.
3. Choose the **Allow a program through Windows Firewall** option and click the **Add port** button.
4. Enter **MySQL** into the **Name** text box and **3306** (or the port of your choice) into the **Port number** text box.

5. Also ensure that the **TCP** protocol radio button is selected.
6. If you wish, you can also limit access to the MySQL server by choosing the **Change scope** button.
7. Confirm your choices by clicking the **OK** button.

Additionally, when running the MySQL Installation Wizard on Windows Vista or newer, ensure that you are logged in as a user with administrative rights.

Note

When using Windows Vista or newer, you may want to disable User Account Control (UAC) before performing the installation. If you do not do so, then MySQL may be identified as a security risk, which will mean that you need to enable MySQL. You can disable the security checking by following these instructions:

1. Open **Control Panel**.
2. Under the **User Accounts and Family Safety**, select **Add or remove user accounts**.
3. Click the **Got to the main User Accounts page** link.
4. Click on **Turn User Account Control on or off**. You may be prompted to provide permission to change this setting. Click **Continue**.
5. Deselect or uncheck the check box next to **Use User Account Control (UAC) to help protect your computer**. Click **OK** to save the setting.

You will need to restart to complete the process. Click **Restart Now** to reboot the machine and apply the changes. You can then follow the instructions below for installing Windows.

5.5.1 Using the MySQL Installation Wizard

MySQL Installation Wizard is an installer for the MySQL server that uses the latest installer technologies for Microsoft Windows. The MySQL Installation Wizard, in combination with the MySQL Configuration Wizard, enables a user to install and configure a MySQL server that is ready for use immediately after installation.

The MySQL Installation Wizard is the standard installer for all MySQL server distributions, version 4.1.5 and higher. Users of previous versions of MySQL need to shut down and remove their existing MySQL installations manually before installing MySQL with the MySQL Installation Wizard. See [Section 5.5.1.6, "Upgrading MySQL with the Installation Wizard"](#), for more information on upgrading from a previous version.

Microsoft has included an improved version of their Microsoft Windows Installer (MSI) in the recent versions of Windows. MSI has become the de-facto standard for application installations on Windows 2000, Windows XP, and Windows Server 2003. The MySQL Installation Wizard makes use of this technology to provide a smoother and more flexible installation process.

The Microsoft Windows Installer Engine was updated with the release of Windows XP; those using a previous version of Windows can reference [this Microsoft Knowledge Base article](#) for information on upgrading to the latest version of the Windows Installer Engine.

In addition, Microsoft has introduced the WiX (Windows Installer XML) toolkit recently. This is the first highly acknowledged Open Source project from Microsoft. We have switched to WiX because it is an Open Source project and it enables us to handle the complete Windows installation process in a flexible manner using scripts.

Improving the MySQL Installation Wizard depends on the support and feedback of users like you. If you find that the MySQL Installation Wizard is lacking some feature important to you, or if you discover a bug, please report it in our bugs database using the instructions given in [How to Report Bugs or Problems](#).

5.5.1.1 Downloading and Starting the MySQL Installation Wizard

The MySQL installation packages can be downloaded from <http://dev.mysql.com/downloads/>. If the package you download is contained within a Zip archive, you need to extract the archive first.

Note

If you are installing on Windows Vista or newer, it is best to open a network port before beginning the installation. To do this, first ensure that you are logged in as an Administrator, go to the [Control Panel](#), and double-click the [Windows Firewall](#) icon. Choose the [Allow a program through Windows Firewall](#) option and click the **Add port** button. Enter `MySQL` into the **Name** text box and `3306` (or the port of your choice) into the **Port number** text box. Also ensure that the **TCP** protocol radio button is selected. If you wish, you can also limit access to the MySQL server by choosing the **Change scope** button. Confirm your choices by clicking the **OK** button. If you do not open a port prior to installation, you cannot configure the MySQL server immediately after installation. Additionally, when running the MySQL Installation Wizard on Windows Vista or newer, ensure that you are logged in as a user with administrative rights.

The process for starting the wizard depends on the contents of the installation package you download. If there is a `setup.exe` file present, double-click it to start the installation process. If there is an `.msi` file present, double-click it to start the installation process.

5.5.1.2 Choosing an Install Type

There are three installation types available: **Typical**, **Complete**, and **Custom**.

The **Typical** installation type installs the MySQL server, the `mysql` command-line client, and the command-line utilities. The command-line clients and utilities include `mysqldump`, `myisamchk`, and several other tools to help you manage the MySQL server.

The **Complete** installation type installs all components included in the installation package. The full installation package includes components such as the embedded server library, the benchmark suite, support scripts, and documentation.

The **Custom** installation type gives you complete control over which packages you wish to install and the installation path that is used. See [Section 5.5.1.3, “The Custom Install Dialog”](#), for more information on performing a custom install.

If you choose the **Typical** or **Complete** installation types and click the **Next** button, you advance to the confirmation screen to verify your choices and begin the installation. If you choose the **Custom** installation type and click the **Next** button, you advance to the custom installation dialog, described in [Section 5.5.1.3, “The Custom Install Dialog”](#).

5.5.1.3 The Custom Install Dialog

If you wish to change the installation path or the specific components that are installed by the MySQL Installation Wizard, choose the **Custom** installation type.

A tree view on the left side of the custom install dialog lists all available components. Components that are not installed have a red **X** icon; components that are installed have a gray icon. To change whether a component is installed, click that component's icon and choose a new option from the drop-down list that appears.

You can change the default installation path by clicking the **Change...** button to the right of the displayed installation path.

After choosing your installation components and installation path, click the **Next** button to advance to the confirmation dialog.

5.5.1.4 The Confirmation Dialog

Once you choose an installation type and optionally choose your installation components, you advance to the confirmation dialog. Your installation type and installation path are displayed for you to review.

To install MySQL if you are satisfied with your settings, click the **Install** button. To change your settings, click the **Back** button. To exit the MySQL Installation Wizard without installing MySQL, click the **Cancel** button.

The final screen of the installer provides a summary of the installation and gives you the option to launch the MySQL Configuration Wizard, which you can use to create a configuration file, install the MySQL service, and configure security settings.

5.5.1.5 Changes Made by MySQL Installation Wizard

Once you click the **Install** button, the MySQL Installation Wizard begins the installation process and makes certain changes to your system which are described in the sections that follow.

Changes to the Registry

The MySQL Installation Wizard creates one Windows registry key in a typical install situation, located in `HKEY_LOCAL_MACHINE\SOFTWARE\MySQL AB`.

The MySQL Installation Wizard creates a key named after the release series of the server that is being installed, such as `MySQL Server 5.5`. It contains two string values, `Location` and `Version`. The `Location` string contains the path to the installation directory. In a default installation it contains `C:\Program Files\MySQL\MySQL Server 5.5\`. The `Version` string contains the release number. For example, for an installation of MySQL Server 5.5.58, the key contains a value of `5.5.58`.

These registry keys are used to help external tools identify the installed location of the MySQL server, preventing a complete scan of the hard-disk to determine the installation path of the MySQL server. The registry keys are not required to run the server, and if you install MySQL using the `noinstall` Zip archive, the registry keys are not created.

Changes to the Start Menu

The MySQL Installation Wizard creates a new entry in the Windows **Start** menu under a common MySQL menu heading named after the release series of MySQL that you have installed. For example, if you install MySQL 5.5, the MySQL Installation Wizard creates a **MySQL Server 5.5** section in the **Start** menu.

The following entries are created within the new **Start** menu section:

- **MySQL Command-Line Client:** This is a shortcut to the `mysql` command-line client and is configured to connect as the `root` user. The shortcut prompts for a `root` user password when you connect.
- **MySQL Server Instance Config Wizard:** This is a shortcut to the MySQL Configuration Wizard. Use this shortcut to configure a newly installed server, or to reconfigure an existing server.
- **MySQL Documentation:** This is a link to the MySQL server documentation that is stored locally in the MySQL server installation directory.

Changes to the File System

The MySQL Installation Wizard by default installs the MySQL 5.5 server to `C:\Program Files\MySQL\MySQL Server 5.5`, where *Program Files* is the default location for applications in your system, and `5.5` is the release series of your MySQL server. This is the recommended location for the MySQL server, replacing the former default location `C:\mysql`.

By default, all MySQL applications are stored in a common directory at `C:\Program Files\MySQL`, where *Program Files* is the default location for applications in your Windows installation. A typical MySQL installation on a developer machine might look like this:

```
C:\Program Files\MySQL\MySQL Server 5.5
C:\Program Files\MySQL\MySQL Workbench 5.1 OSS
```

This approach makes it easier to manage and maintain all MySQL applications installed on a particular system.

The default location of the data directory is the `AppData` directory configured for the user that installed the MySQL application.

5.5.1.6 Upgrading MySQL with the Installation Wizard

The MySQL Installation Wizard can perform server upgrades automatically using the upgrade capabilities of MSI. That means you do not need to remove a previous installation manually before installing a new release. The installer automatically shuts down and removes the previous MySQL service before installing the new version.

Automatic upgrades are available only when upgrading between installations that have the same major and minor version numbers. For example, you can upgrade automatically from MySQL 5.5.5 to MySQL 5.5.6, but not from MySQL 5.1 to MySQL 5.5.

See [Section 5.10, "Upgrading MySQL on Windows"](#).

5.5.2 Automating MySQL Installation on Microsoft Windows Using the MSI Package

The Microsoft Installer (MSI) supports both a *quiet* and a *passive* mode that can be used to install MySQL automatically without requiring intervention. You can use this either in scripts to automatically install MySQL or through a terminal connection such as Telnet where you do not have access to the standard Windows user interface. The MSI packages can also be used in combination with Microsoft's Group Policy system (part of Windows Server 2003 and Windows Server 2008) to install MySQL across multiple machines.

To install MySQL from one of the MSI packages automatically from the command line (or within a script), you need to use the `msiexec.exe` tool. For example, to perform a quiet installation (which shows no dialog boxes or progress):

```
shell> msiexec /i mysql-5.5.58.msi /quiet
```


The `/i` indicates that you want to perform an installation. The `/quiet` option indicates that you want no interactive elements.

To provide a dialog box showing the progress during installation, and the dialog boxes providing information on the installation and registration of MySQL, use `/passive` mode instead of `/quiet`:

```
shell> msiexec /i mysql-5.5.58.msi /passive
```

Regardless of the mode of the installation, installing the package in this manner performs a 'Typical' installation, and installs the default components into the standard location.

You can also use this method to uninstall MySQL by using the `/uninstall` or `/x` options:

```
shell> msiexec /x mysql-5.5.58.msi /uninstall
```

To install MySQL and configure a MySQL instance from the command line, see [Section 5.6.13, “MySQL Server Instance Config Wizard: Creating an Instance from the Command Line”](#).

For information on using MSI packages to install software automatically using Group Policy, see [How to use Group Policy to remotely install software in Windows Server 2003](#).

5.5.3 Removing MySQL When Installed from the MSI Package

To uninstall MySQL when you installed it using the MSI package, you must use the **Add/Remove Programs** tool within **Control Panel**. To do this:

1. Right-click the **start** menu and choose **Control Panel**.
2. If the Control Panel is set to category mode (you will see **Pick a category** at the top of the **Control Panel** window), double-click **Add or Remove Programs**. If the Control is set to classic mode, double-click the **Add or Remove Programs** icon.
3. Find MySQL in the list of installed software. MySQL Server is installed against release series numbers (MySQL 5.1, MySQL 5.5, etc.). Select the version that you want to remove and click **Remove**.
4. You will be prompted to confirm the removal. Click **Yes** to remove MySQL.

When MySQL is removed using this method, only the installed components are removed. Any database information (including the tables and data), import or export files, log files, and binary logs produced during execution are kept in their configured location.

If you try to install MySQL again the information will be retained and you will be prompted to enter the password configured with the original installation.

If you want to delete MySQL completely:

- Delete the associated data directory. On Windows XP and Windows Server 2003, the default data directory is the configured AppData directory, which is `C:\Documents and Settings\All Users\Application Data\MySQL` by default.
- On Windows 7 and Windows Server 2008, the default data directory location is `C:\ProgramData\MySQL`.

Note

The `C:\ProgramData` directory is hidden by default. You must change your folder options to view the hidden file. Choose **Organize, Folder and search options, Show hidden folders**.

5.6 MySQL Server Instance Configuration Wizard

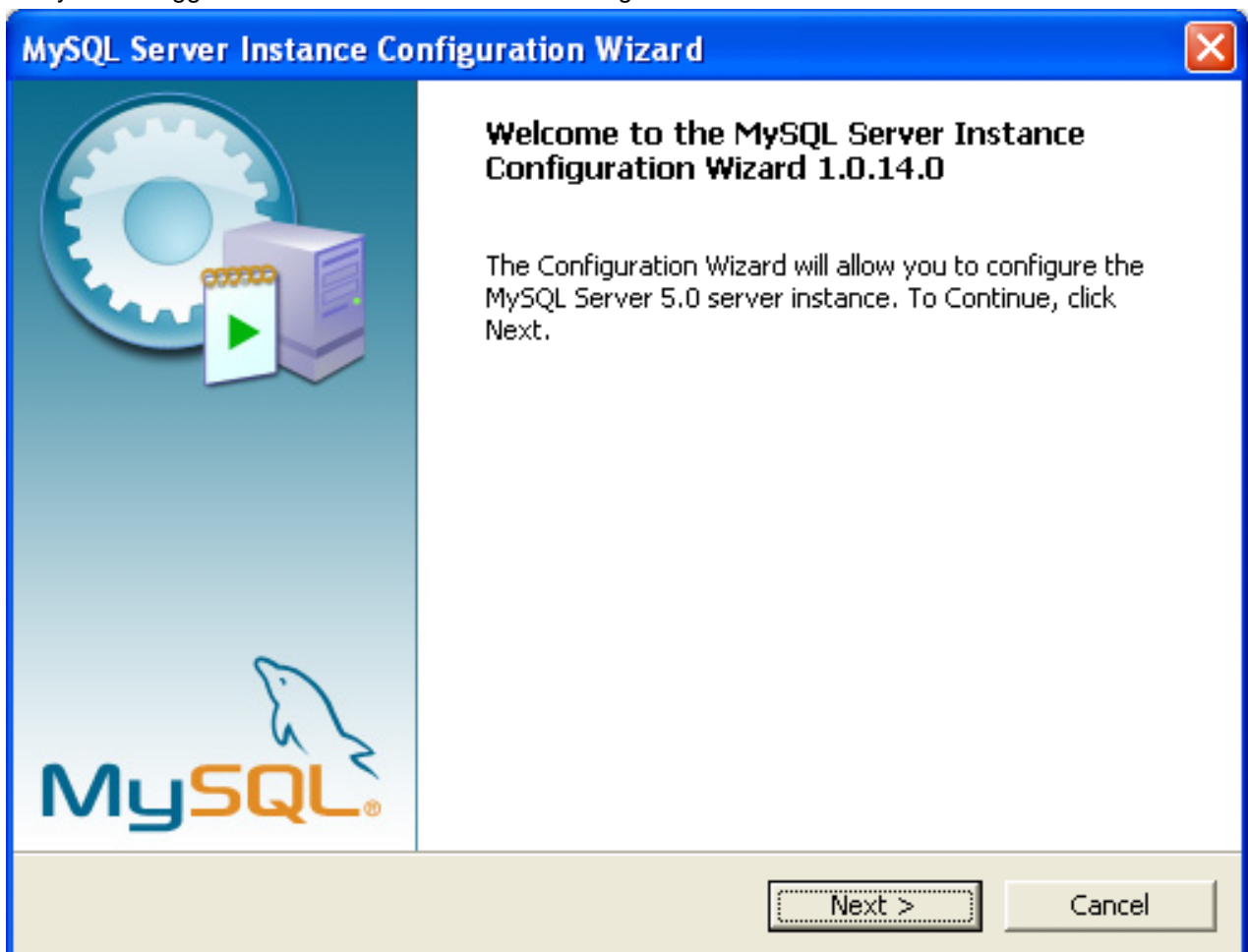
The MySQL Server Instance Configuration Wizard helps automate the process of configuring your server. It creates a custom MySQL configuration file (`my.ini` or `my.cnf`) by asking you a series of questions and then applying your responses to a template to generate the configuration file that is tuned to your installation.

The MySQL Server Instance Configuration Wizard is included with the MySQL 5.5 server. The MySQL Server Instance Configuration Wizard is only available for Windows.

5.6.1 Starting the MySQL Server Instance Configuration Wizard

The MySQL Server Instance Configuration Wizard is normally started as part of the installation process. You should only need to run the MySQL Server Instance Configuration Wizard again when you need to change the configuration parameters of your server.

If you chose not to open a port prior to installing MySQL on Windows Vista or newer, you can choose to use the MySQL Server Configuration Wizard after installation. However, you must open a port in the Windows Firewall. To do this see the instructions given in [Section 5.5.1.1, "Downloading and Starting the MySQL Installation Wizard"](#). Rather than opening a port, you also have the option of adding MySQL as a program that bypasses the Windows Firewall. One or the other option is sufficient—you need not do both. Additionally, when running the MySQL Server Configuration Wizard on Windows Vista or newer, ensure that you are logged in as a user with administrative rights.



You can launch the MySQL Configuration Wizard by clicking the **MySQL Server Instance Config Wizard** entry in the **MySQL** section of the Windows **Start** menu.

Alternatively, you can navigate to the `bin` directory of your MySQL installation and launch the `MySQLInstanceConfig.exe` file directly.

The MySQL Server Instance Configuration Wizard places the `my.ini` file in the installation directory for the MySQL server. This helps associate configuration files with particular server instances.

To ensure that the MySQL server knows where to look for the `my.ini` file, an argument similar to this is passed to the MySQL server as part of the service installation:

```
--defaults-file="C:\Program Files\MySQL\MySQL Server 5.5\my.ini"
```

Here, `C:\Program Files\MySQL\MySQL Server 5.5` is replaced with the installation path to the MySQL Server. The `--defaults-file` option instructs the MySQL server to read the specified file for configuration options when it starts.

Apart from making changes to the `my.ini` file by running the MySQL Server Instance Configuration Wizard again, you can modify it by opening it with a text editor and making any necessary changes. You can also modify the server configuration with the <http://www.mysql.com/products/administrator/> utility. For more information about server configuration, see [Server Command Options](#).

MySQL clients and utilities such as the `mysql` and `mysqldump` command-line clients are not able to locate the `my.ini` file located in the server installation directory. To configure the client and utility applications, create a new `my.ini` file in the Windows installation directory (for example, `C:\WINDOWS`).

Under Windows Server 2003, Windows Server 2000, Windows XP, and Windows Vista, MySQL Server Instance Configuration Wizard will configure MySQL to work as a Windows service. To start and stop MySQL you use the [Services](#) application that is supplied as part of the Windows Administrator Tools.

5.6.2 Choosing a Maintenance Option

If the MySQL Server Instance Configuration Wizard detects an existing configuration file, you have the option of either reconfiguring your existing server, or removing the server instance by deleting the configuration file and stopping and removing the MySQL service.

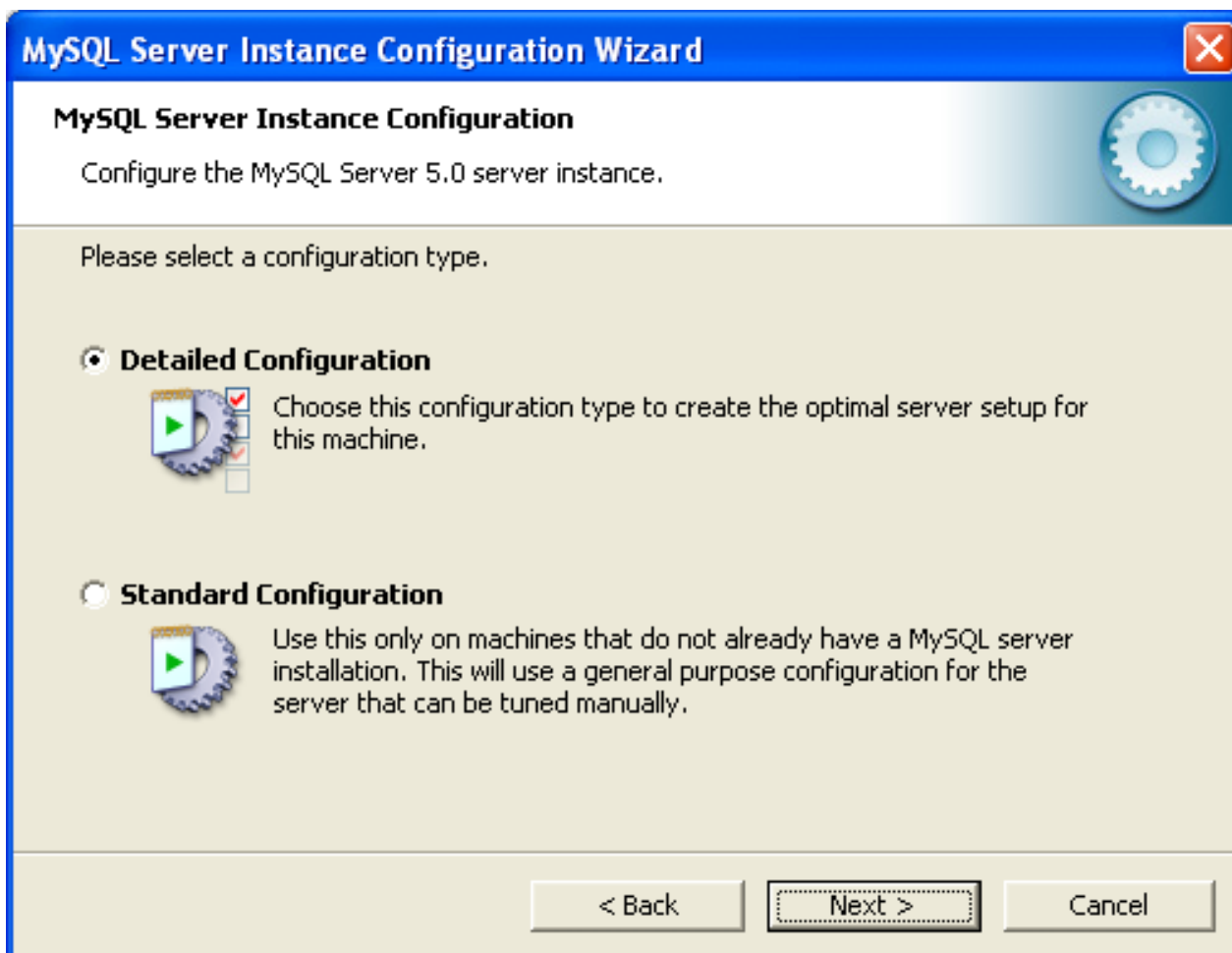
To reconfigure an existing server, choose the **Re-configure Instance** option and click the **Next** button. Any existing configuration file is not overwritten, but renamed (within the same directory) using a timestamp (Windows) or sequential number (Linux). To remove the existing server instance, choose the **Remove Instance** option and click the **Next** button.

If you choose the **Remove Instance** option, you advance to a confirmation window. Click the **Execute** button. The MySQL Server Configuration Wizard stops and removes the MySQL service, and then deletes the configuration file. The server installation and its `data` folder are not removed.

If you choose the **Re-configure Instance** option, you advance to the **Configuration Type** dialog where you can choose the type of installation that you wish to configure.

5.6.3 Choosing a Configuration Type

When you start the MySQL Server Instance Configuration Wizard for a new MySQL installation, or choose the **Re-configure Instance** option for an existing installation, you advance to the **Configuration Type** dialog.



There are two configuration types available: **Detailed Configuration** and **Standard Configuration**. The **Standard Configuration** option is intended for new users who want to get started with MySQL quickly without having to make many decisions about server configuration. The **Detailed Configuration** option is intended for advanced users who want more fine-grained control over server configuration.

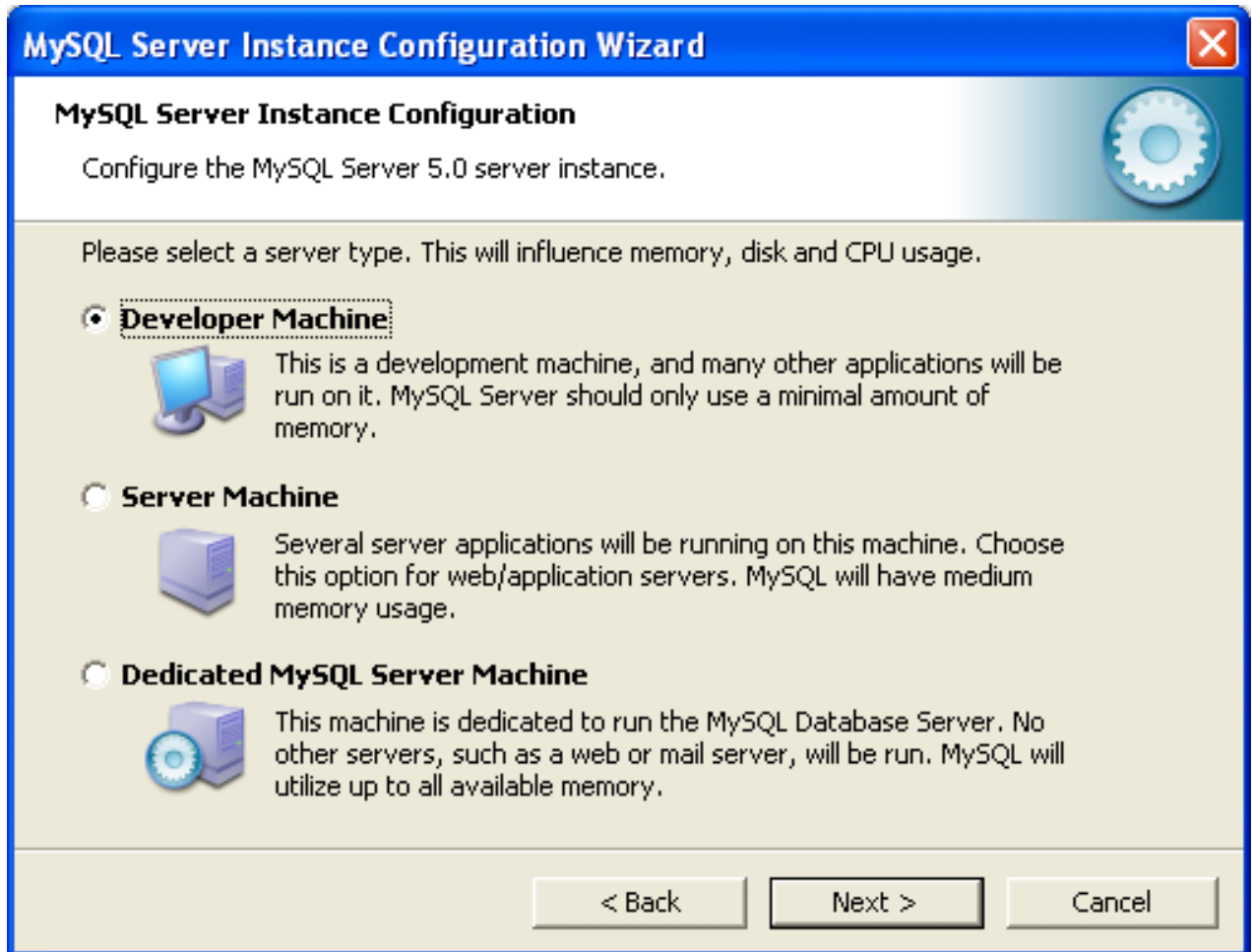
If you are new to MySQL and need a server configured as a single-user developer machine, the **Standard Configuration** should suit your needs. Choosing the **Standard Configuration** option causes the MySQL Configuration Wizard to set all configuration options automatically with the exception of **Service Options** and **Security Options**.

The **Standard Configuration** sets options that may be incompatible with systems where there are existing MySQL installations. If you have an existing MySQL installation on your system in addition to the installation you wish to configure, the **Detailed Configuration** option is recommended.

To complete the **Standard Configuration**, please refer to the sections on **Service Options** and **Security Options** in [Section 5.6.10, “The Service Options Dialog”](#), and [Section 5.6.11, “The Security Options Dialog”](#), respectively.

5.6.4 The Server Type Dialog

There are three different server types available to choose from. The server type that you choose affects the decisions that the MySQL Server Instance Configuration Wizard makes with regard to memory, disk, and processor usage.



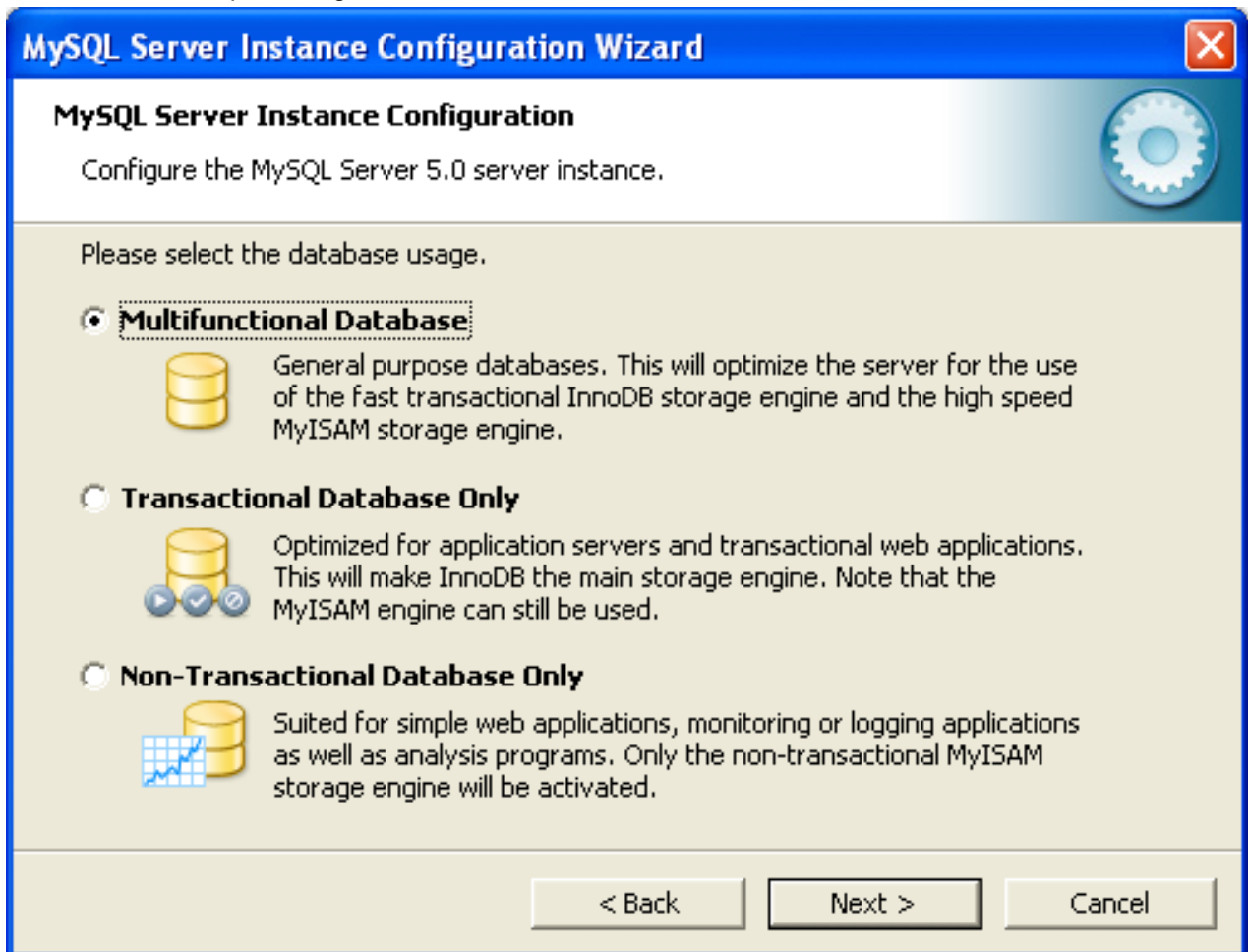
- **Developer Machine:** Choose this option for a typical desktop workstation where MySQL is intended only for personal use. It is assumed that many other desktop applications are running. The MySQL server is configured to use minimal system resources.
- **Server Machine:** Choose this option for a server machine where the MySQL server is running alongside other server applications such as FTP, email, and Web servers. The MySQL server is configured to use a moderate portion of the system resources.
- **Dedicated MySQL Server Machine:** Choose this option for a server machine that is intended to run only the MySQL server. It is assumed that no other applications are running. The MySQL server is configured to use all available system resources.

Note

By selecting one of the preconfigured configurations, the values and settings of various options in your `my.cnf` or `my.ini` will be altered accordingly. The default values and options as described in the reference manual may therefore be different to the options and values that were created during the execution of the configuration wizard.

5.6.5 The Database Usage Dialog

The **Database Usage** dialog enables you to indicate the storage engines that you expect to use when creating MySQL tables. The option you choose determines whether the [InnoDB](#) storage engine is available and what percentage of the server resources are available to [InnoDB](#).



- **Multifunctional Database:** This option enables both the [InnoDB](#) and [MyISAM](#) storage engines and divides resources evenly between the two. This option is recommended for users who use both storage engines on a regular basis.
- **Transactional Database Only:** This option enables both the [InnoDB](#) and [MyISAM](#) storage engines, but dedicates most server resources to the [InnoDB](#) storage engine. This option is recommended for users who use [InnoDB](#) almost exclusively and make only minimal use of [MyISAM](#).
- **Non-Transactional Database Only:** This option disables the [InnoDB](#) storage engine completely and dedicates all server resources to the [MyISAM](#) storage engine. This option is recommended for users who do not use [InnoDB](#).

The Configuration Wizard uses a template to generate the server configuration file. The **Database Usage** dialog sets one of the following option strings:

```
Multifunctional Database:      MIXED
Transactional Database Only:  INNODB
Non-Transactional Database Only: MYISAM
```

When these options are processed through the default template (my-template.ini) the result is:

```
Multifunctional Database:
default-storage-engine=InnoDB
_myisam_pct=50

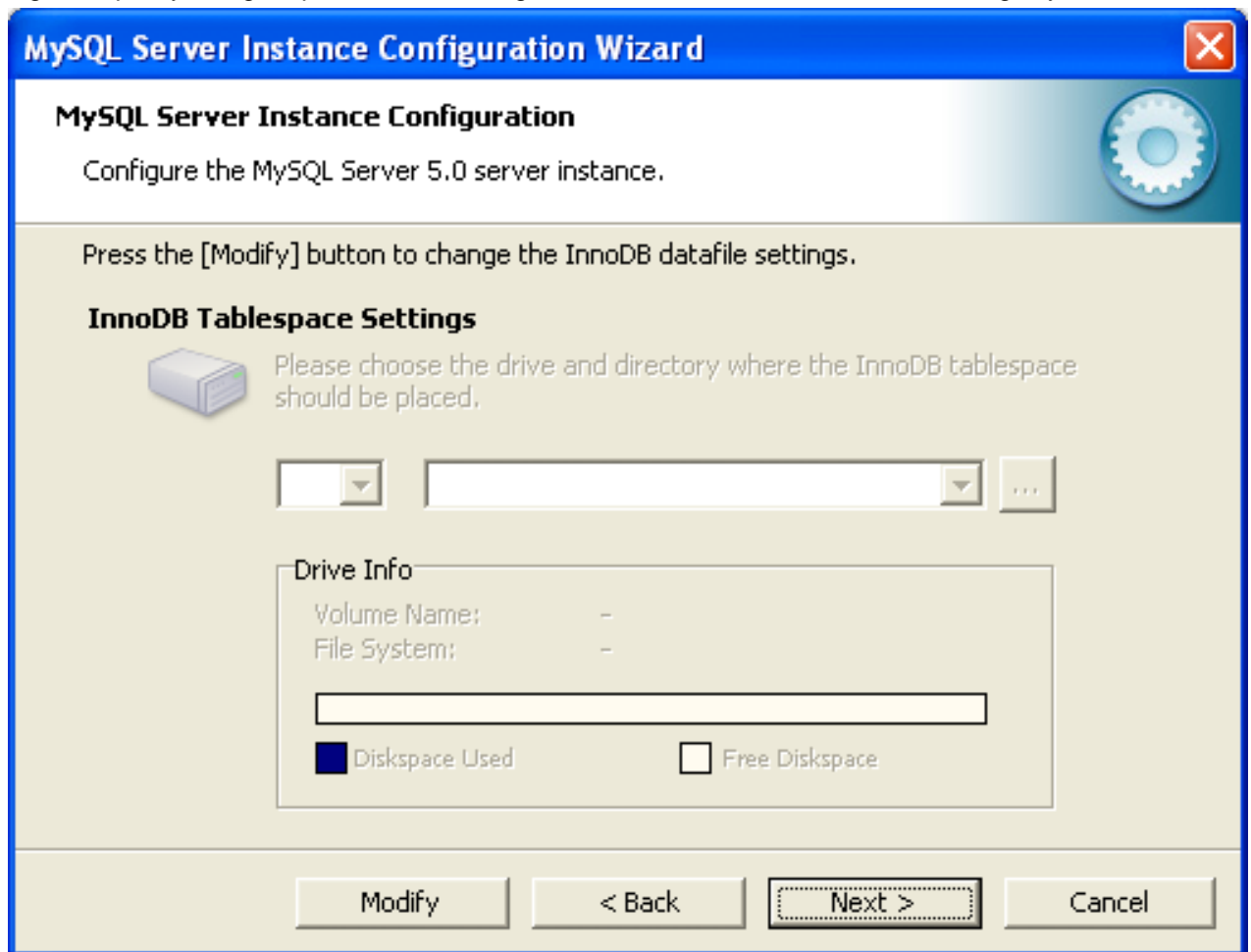
Transactional Database Only:
default-storage-engine=InnoDB
_myisam_pct=5

Non-Transactional Database Only:
default-storage-engine=MyISAM
_myisam_pct=100
skip-innodb
```

The `_myisam_pct` value is used to calculate the percentage of resources dedicated to `MyISAM`. The remaining resources are allocated to `InnoDB`.

5.6.6 The InnoDB Tablespace Dialog

Some users may want to locate the `InnoDB` tablespace files in a different location than the MySQL server data directory. Placing the tablespace files in a separate location can be desirable if your system has a higher capacity or higher performance storage device available, such as a RAID storage system.

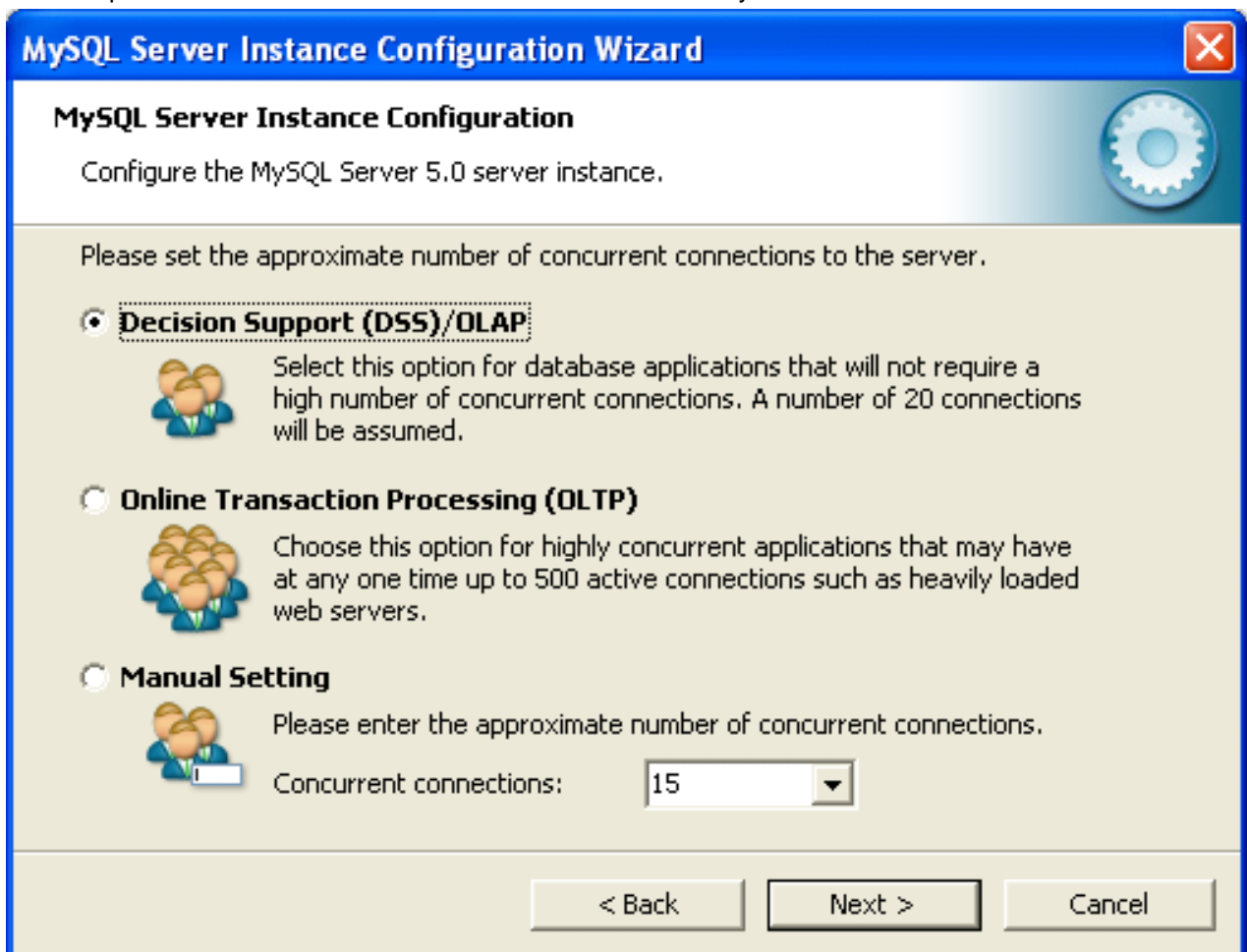


To change the default location for the [InnoDB](#) tablespace files, choose a new drive from the drop-down list of drive letters and choose a new path from the drop-down list of paths. To create a custom path, click the ... button.

If you are modifying the configuration of an existing server, you must click the **Modify** button before you change the path. In this situation you must move the existing tablespace files to the new location manually before starting the server.

5.6.7 The Concurrent Connections Dialog

To prevent the server from running out of resources, it is important to limit the number of concurrent connections to the MySQL server that can be established. The **Concurrent Connections** dialog enables you to choose the expected usage of your server, and sets the limit for concurrent connections accordingly. It is also possible to set the concurrent connection limit manually.

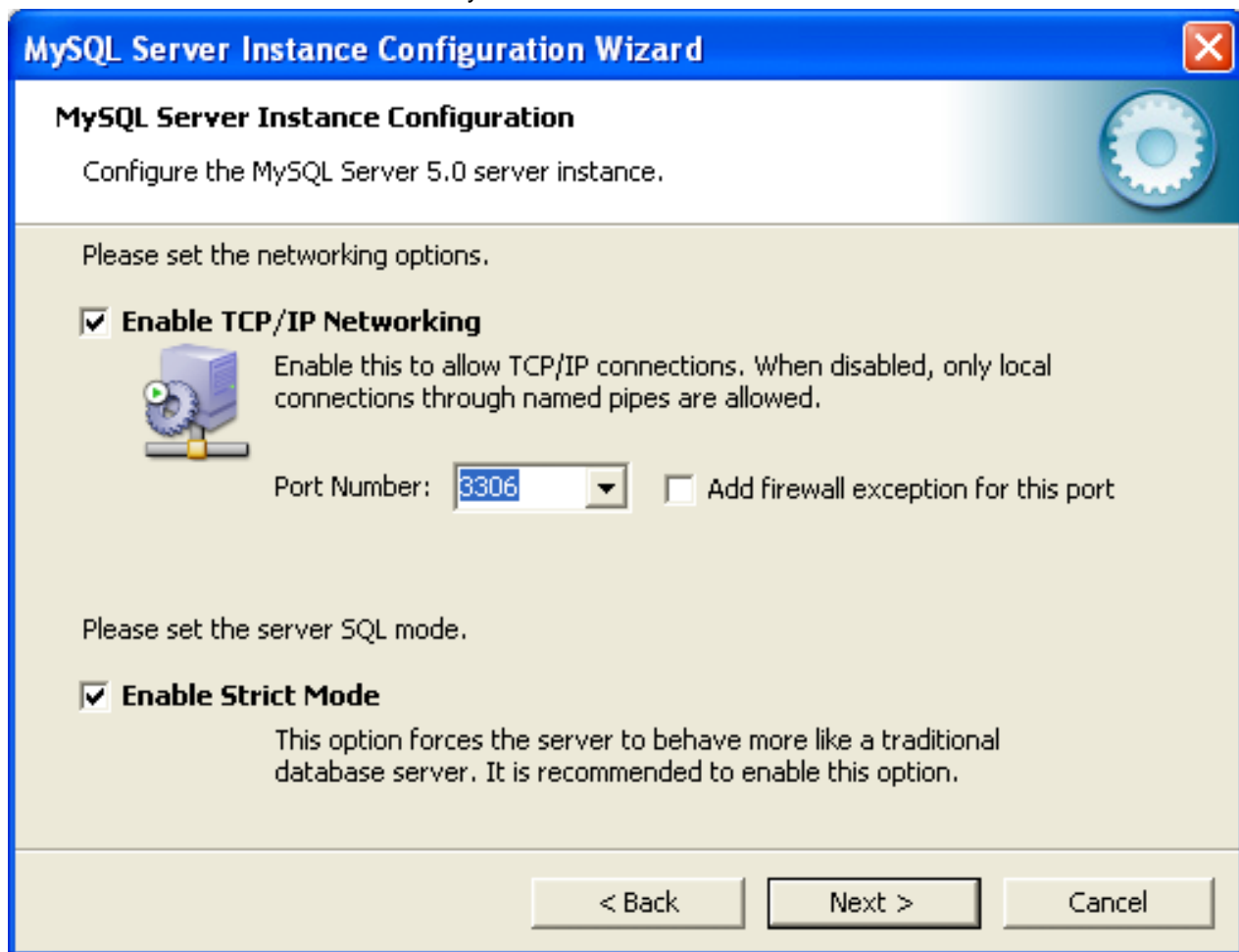


- **Decision Support (DSS)/OLAP:** Choose this option if your server does not require a large number of concurrent connections. The maximum number of connections is set at 100, with an average of 20 concurrent connections assumed.
- **Online Transaction Processing (OLTP):** Choose this option if your server requires a large number of concurrent connections. The maximum number of connections is set at 500.

- **Manual Setting:** Choose this option to set the maximum number of concurrent connections to the server manually. Choose the number of concurrent connections from the drop-down box provided, or enter the maximum number of connections into the drop-down box if the number you desire is not listed.

5.6.8 The Networking and Strict Mode Options Dialog

Use the **Networking Options** dialog to enable or disable TCP/IP networking and to configure the port number that is used to connect to the MySQL server.



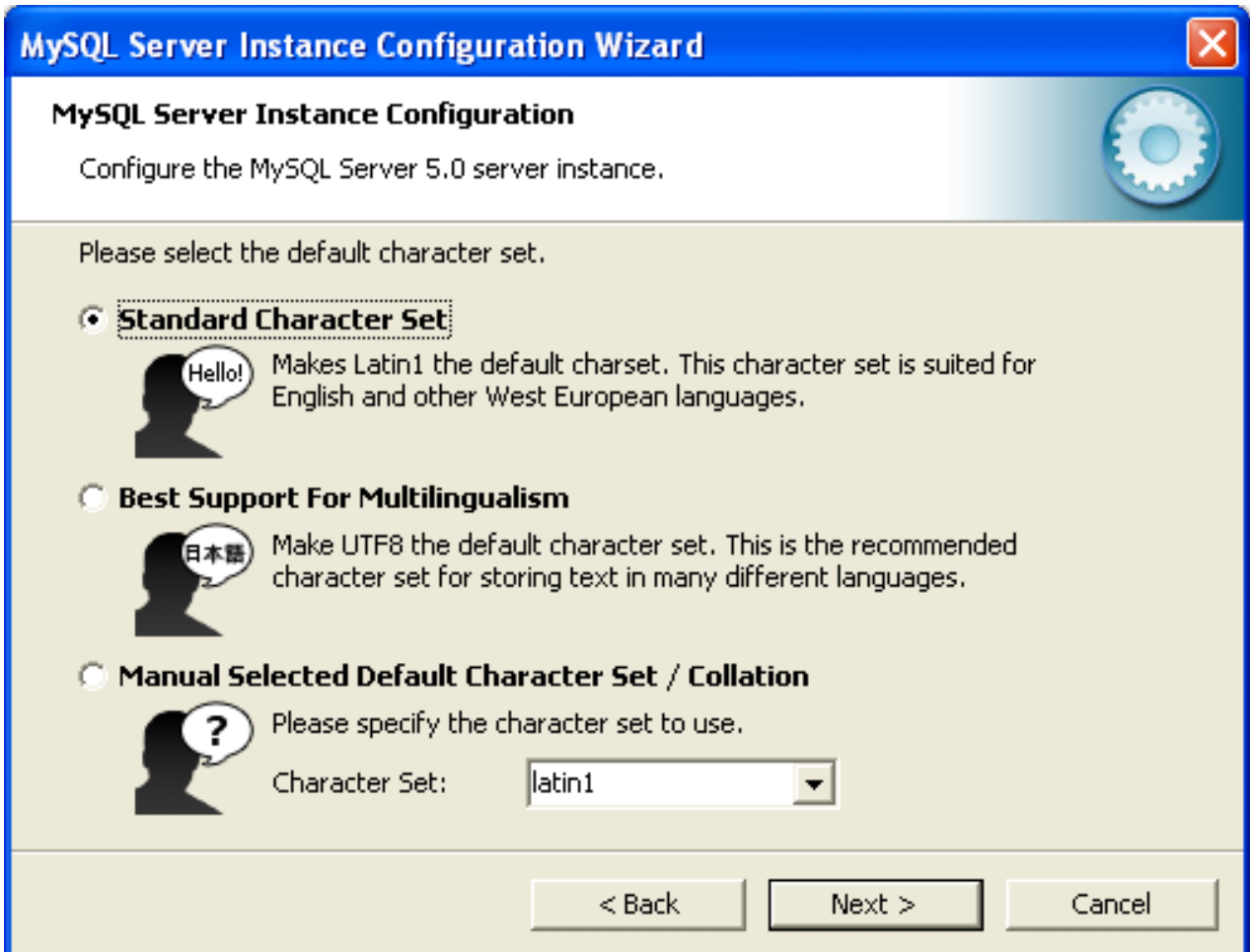
TCP/IP networking is enabled by default. To disable TCP/IP networking, uncheck the box next to the **Enable TCP/IP Networking** option.

Port 3306 is used by default. To change the port used to access MySQL, choose a new port number from the drop-down box or type a new port number directly into the drop-down box. If the port number you choose is in use, you are prompted to confirm your choice of port number.

Set the **Server SQL Mode** to either enable or disable strict mode. Enabling strict mode (default) makes MySQL behave more like other database management systems. *If you run applications that rely on MySQL's old "forgiving" behavior, make sure to either adapt those applications or to disable strict mode.* For more information about strict mode, see [Server SQL Modes](#).

5.6.9 The Character Set Dialog

The MySQL server supports multiple character sets and it is possible to set a default server character set that is applied to all tables, columns, and databases unless overridden. Use the **Character Set** dialog to change the default character set of the MySQL server.



- **Standard Character Set:** Choose this option if you want to use `latin1` as the default server character set. `latin1` is used for English and many Western European languages.
- **Best Support For Multilingualism:** Choose this option if you want to use `utf8` as the default server character set. This is a Unicode character set that can store characters from many different languages.
- **Manual Selected Default Character Set / Collation:** Choose this option if you want to pick the server's default character set manually. Choose the desired character set from the provided drop-down list.

5.6.10 The Service Options Dialog

On Windows platforms, the MySQL server can be installed as a Windows service. When installed this way, the MySQL server can be started automatically during system startup, and even restarted automatically by Windows in the event of a service failure.

The MySQL Server Instance Configuration Wizard installs the MySQL server as a service by default, using the service name `MySQL`. If you do not wish to install the service, uncheck the box next to the **Install As**

Windows Service option. You can change the service name by picking a new service name from the drop-down box provided or by entering a new service name into the drop-down box.

Note

Service names can include any legal character except forward (/) or backward (\) slashes, and must be less than 256 characters long.

Warning

If you are installing multiple versions of MySQL onto the same machine, you *must* choose a different service name for each version that you install. If you do not choose a different service for each installed version then the service manager information will be inconsistent and this will cause problems when you try to uninstall a previous version.

If you have already installed multiple versions using the same service name, you must manually edit the contents of the [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services](#) parameters within the Windows registry to update the association of the service name with the correct server version.

Typically, when installing multiple versions you create a service name based on the version information. For example, you might install MySQL 5.x as `mysql5`, or specific versions such as MySQL 5.5.0 as `mysql50500`.

To install the MySQL server as a service but not have it started automatically at startup, uncheck the box next to the **Launch the MySQL Server Automatically** option.

5.6.11 The Security Options Dialog

The content of the security options portion of the MySQL Server Instance Configuration Wizard will depend on whether this is a new installation, or modifying an existing installation.

- **Setting the root password for a new installation**

*It is strongly recommended that you set a `root` password for your MySQL server, and the MySQL Server Instance Config Wizard requires by default that you do so. If you do not wish to set a `root` password, uncheck the box next to the **Modify Security Settings** option.*

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Please set the security options.

Modify Security Settings

 New root password: Enter the root password.

Confirm: Retype the password.

Enable root access from remote machines

Create An Anonymous Account

 This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

< Back Next > Cancel

- To set the `root` password, enter the desired password into both the **New root password** and **Confirm** boxes.

Setting the root password for an existing installation

If you are modifying the configuration of an existing configuration, or you are installing an upgrade and the MySQL Server Instance Configuration Wizard has detected an existing MySQL system, then you must enter the existing password for `root` before changing the configuration information.

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.1 server instance.

Please set the security options.

Modify Security Settings

 Current root password: Enter the current password.

 New root password: Enter the root password.

Confirm: Retype the password.

Enable root access from remote machines

Create An Anonymous Account

 This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

< Back Next > Cancel

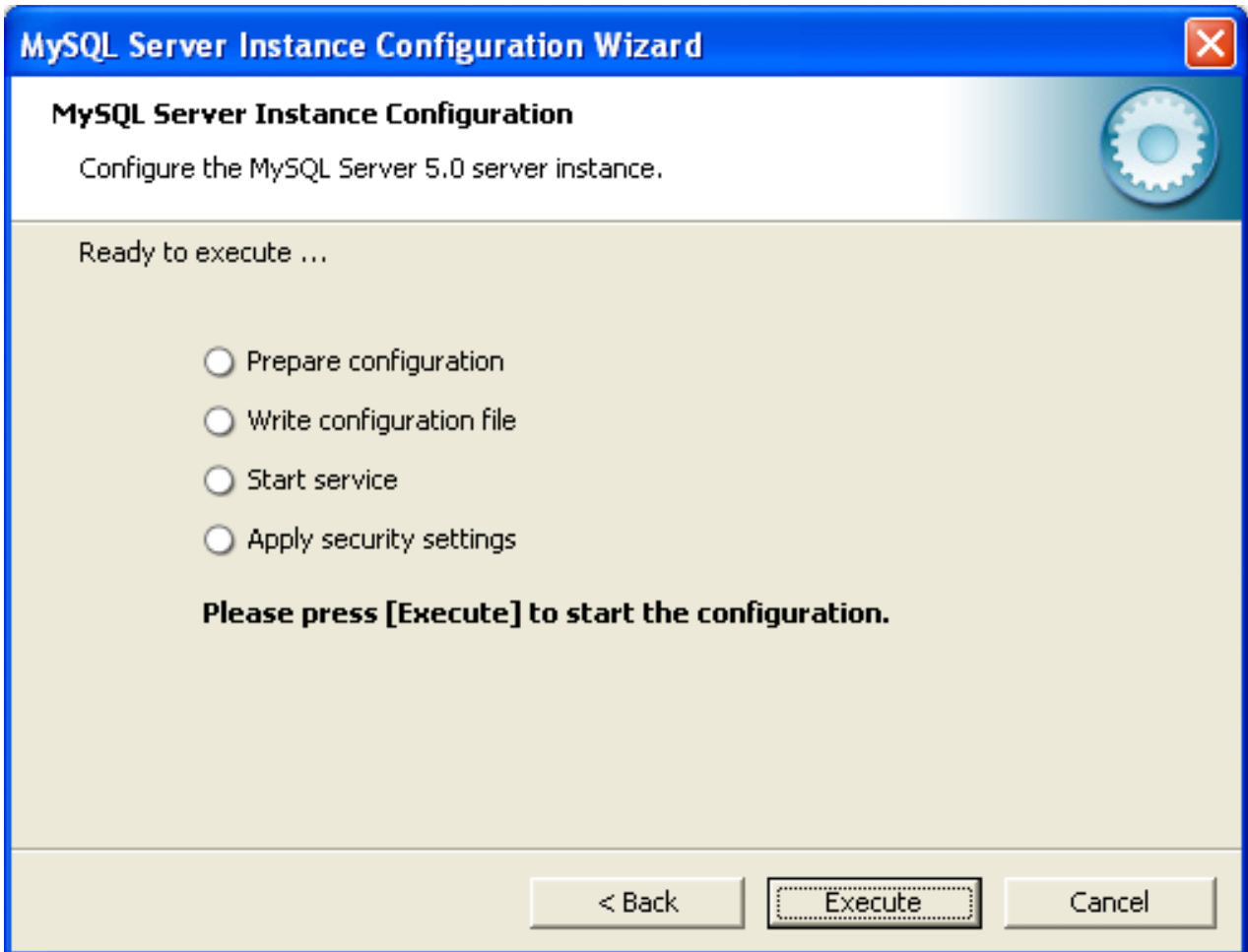
If you want to change the current `root` password, enter the desired new password into both the **New root password** and **Confirm** boxes.

To permit `root` logins from across the network, check the box next to the **Enable root access from remote machines** option. This decreases the security of your `root` account.

To create an anonymous user account, check the box next to the **Create An Anonymous Account** option. Creating an anonymous account can decrease server security and cause login and permission difficulties. For this reason, it is not recommended.

5.6.12 The Confirmation Dialog

The final dialog in the MySQL Server Instance Configuration Wizard is the **Confirmation Dialog**. To start the configuration process, click the **Execute** button. To return to a previous dialog, click the **Back** button. To exit the MySQL Server Instance Configuration Wizard without configuring the server, click the **Cancel** button.



After you click the **Execute** button, the MySQL Server Instance Configuration Wizard performs a series of tasks and displays the progress onscreen as the tasks are performed.

The MySQL Server Instance Configuration Wizard first determines configuration file options based on your choices using a template prepared by MySQL developers and engineers. This template is named `my-template.ini` and is located in your server installation directory.

The MySQL Configuration Wizard then writes these options to the corresponding configuration file.

If you chose to create a service for the MySQL server, the MySQL Server Instance Configuration Wizard creates and starts the service. If you are reconfiguring an existing service, the MySQL Server Instance Configuration Wizard restarts the service to apply your configuration changes.

If you chose to set a `root` password, the MySQL Configuration Wizard connects to the server, sets your new `root` password, and applies any other security settings you may have selected.

After the MySQL Server Instance Configuration Wizard has completed its tasks, it displays a summary. Click the **Finish** button to exit the MySQL Server Configuration Wizard.

5.6.13 MySQL Server Instance Config Wizard: Creating an Instance from the Command Line

In addition to using the GUI interface to the MySQL Server Instance Config Wizard, you can also create instances automatically from the command line.

To use the MySQL Server Instance Config Wizard on the command line, you need to use the `MySQLInstanceConfig.exe` command that is installed with MySQL in the `bin` directory within the installation directory. `MySQLInstanceConfig.exe` takes a number of command-line arguments that set the properties that would normally be selected through the GUI interface, and then creates a new configuration file (`my.ini`) by combining these selections with a template configuration file to produce the working configuration file.

The main command line options are provided in the table below. Some of the options are required, while some options are optional.

Table 5.3 MySQL Server Instance Config Wizard Command Line Options

Option	Description
Required Parameters	
<code>-nPRODUCTNAME</code>	The name of the instance when installed
<code>-pPATH</code>	Path of the base directory for installation. This is equivalent to the directory when using the <code>basedir</code> configuration parameter
<code>-vVERSION</code>	The version tag to use for this installation
Action to Perform	
<code>-i</code>	Install an instance
<code>-r</code>	Remove an instance
<code>-s</code>	Stop an existing instance
<code>-q</code>	Perform the operation quietly
<code>-lFILENAME</code>	Save the installation progress in a logfile
Config File to Use	
<code>-tFILENAME</code>	Path to the template config file that will be used to generate the installed configuration file
<code>-cFILENAME</code>	Path to a config file to be generated

The `-t` and `-c` options work together to set the configuration parameters for a new instance. The `-t` option specifies the template configuration file to use as the basic configuration, which are then merged with the configuration parameters generated by the MySQL Server Instance Config Wizard into the configuration file specified by the `-c` option.

A sample template file, `my-template.ini` is provided in the toplevel MySQL installation directory. The file contains elements that are replaced automatically by the MySQL Server Instance Config Wizard during configuration.

If you specify a configuration file that already exists, the existing configuration file will be saved in the file with the original, with the date and time added. For example, the `mysql.ini` will be copied to `mysql 2009-10-27 1646.ini.bak`.

The parameters that you can specify on the command line are listed in the table below.

Table 5.4 MySQL Server Instance Config Wizard Parameters

Parameter	Description
<code>ServiceName=\$</code>	Specify the name of the service to be created
<code>AddBinToPath={yes no}</code>	Specifies whether to add the binary directory of MySQL to the standard <code>PATH</code> environment variable

Parameter	Description
<code>ServerType={DEVELOPMENT SERVER DEDICATED}</code>	Specify the server type. For more information, see Section 5.6.4, “The Server Type Dialog”
<code>DatabaseType={MIXED INNODB MYISAM}</code>	Specify the default database type. For more information, see Section 5.6.5, “The Database Usage Dialog”
<code>ConnectionUsage={DSS OLTP}</code>	Specify the type of connection support, this automates the setting for the number of concurrent connections (see the <code>ConnectionCount</code> parameter). For more information, see Section 5.6.7, “The Concurrent Connections Dialog”
<code>ConnectionCount=#</code>	Specify the number of concurrent connections to support. For more information, see Section 5.6.4, “The Server Type Dialog”
<code>SkipNetworking={yes no}</code>	Specify whether network support should be supported. Specifying <code>yes</code> disables network access altogether
<code>Port=#</code>	Specify the network port number to use for network connections. For more information, see Section 5.6.8, “The Networking and Strict Mode Options Dialog”
<code>StrictMode={yes no}</code>	Specify whether to use the <code>strict</code> SQL mode. For more information, see Section 5.6.8, “The Networking and Strict Mode Options Dialog”
<code>Charset=\$</code>	Specify the default character set. For more information, see Section 5.6.9, “The Character Set Dialog”
<code>RootPassword=\$</code>	Specify the root password
<code>RootCurrentPassword=\$</code>	Specify the current root password then stopping or reconfiguring an existing service

Note

When specifying options on the command line, you can enclose the entire command-line option and the value you are specifying using double quotation marks. This enables you to use spaces in the options. For example, `"-cC: \mysql.ini"`.

The following command installs a MySQL Server 5.5 instance from the directory `C:\Program Files\MySQL\MySQL Server 5.5` using the service name `MySQL55` and setting the root password to 1234.

```
shell> MySQLInstanceConfig.exe -i -q "-lC:\mysql_install_log.txt" »
"-nMySQL Server 5.5" "-pC:\Program Files\MySQL\MySQL Server 5.5" -v5.5.58 »
"-tmy-template.ini" "-cC:\mytest.ini" ServerType=DEVELOPMENT DatabaseType=MIXED »
ConnectionUsage=DSS Port=3311 ServiceName=MySQL55 RootPassword=1234
```

In the above example, a log file will be generated in `mysql_install_log.txt` containing the information about the instance creation process. The log file generated by the above example is shown below:

```
Welcome to the MySQL Server Instance Configuration Wizard 1.0.16.0
Date: 2009-10-27 17:07:21

Installing service ...

Product Name:      MySQL Server 5.5
Version:           5.5.58
Installation Path: C:\Program Files\MySQL\MySQL Server 5.5\

Creating configuration file C:\mytest.ini using template my-template.ini.
```



```
Options:
DEVELOPMENT
MIXED
DSS
STRICTMODE

Variables:
port: 3311
default-character-set: latin1
basedir: "C:/Program Files/MySQL/MySQL Server 5.5/"
datadir: "C:/Program Files/MySQL/MySQL Server 5.5/Data/"

Creating Windows service entry.
Service name: "MySQL55"
Parameters: "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqld" --defaults-file="C:\mytest.ini" MySQL55
Windows service MySQL55 installed.
```

When using the command line, the return values in the following table indicate an error performing the specified option.

Table 5.5 Return Value from MySQL Server Instance Config Wizard

Value	Description
2	Configuration template file cannot be found
3	The Windows service entry cannot be created
4	Could not connect to the Service Control Manager
5	The MySQL service cannot be started
6	The MySQL service cannot be stopped
7	The security settings cannot be applied
8	The configuration file cannot be written
9	The Windows service entry cannot be removed

You can perform an installation of MySQL automatically using the MSI package. For more information, see [Section 5.5.2, "Automating MySQL Installation on Microsoft Windows Using the MSI Package"](#).

5.7 Installing MySQL on Microsoft Windows Using a noinstall Zip Archive

Users who are installing from the `noinstall` package can use the instructions in this section to manually install MySQL. The process for installing MySQL from a Zip archive is as follows:

1. Extract the archive to the desired install directory
2. Create an option file
3. Choose a MySQL server type
4. Start the MySQL server
5. Secure the default user accounts

This process is described in the sections that follow.

5.7.1 Extracting the Install Archive

To install MySQL manually, do the following:

1. If you are upgrading from a previous version please refer to [Section 5.10, “Upgrading MySQL on Windows”](#), before beginning the upgrade process.
2. Make sure that you are logged in as a user with administrator privileges.
3. Choose an installation location. Traditionally, the MySQL server is installed in `C:\mysql`. The MySQL Installation Wizard installs MySQL under `C:\Program Files\MySQL`. If you do not install MySQL at `C:\mysql`, you must specify the path to the install directory during startup or in an option file. See [Section 5.7.2, “Creating an Option File”](#).

Note

The MySQL Installer installs MySQL under `C:\Program Files\MySQL`.

4. Extract the install archive to the chosen installation location using your preferred Zip archive tool. Some tools may extract the archive to a folder within your chosen installation location. If this occurs, you can move the contents of the subfolder into the chosen installation location.

5.7.2 Creating an Option File

If you need to specify startup options when you run the server, you can indicate them on the command line or place them in an option file. For options that are used every time the server starts, you may find it most convenient to use an option file to specify your MySQL configuration. This is particularly true under the following circumstances:

- The installation or data directory locations are different from the default locations (`C:\Program Files\MySQL\MySQL Server 5.5` and `C:\Program Files\MySQL\MySQL Server 5.5\data`).
- You need to tune the server settings, such as memory, cache, or InnoDB configuration information.

When the MySQL server starts on Windows, it looks for option files in several locations, such as the Windows directory, `C:\`, and the MySQL installation directory (for the full list of locations, see [Using Option Files](#)). The Windows directory typically is named something like `C:\WINDOWS`. You can determine its exact location from the value of the `WINDIR` environment variable using the following command:

```
C:\> echo %WINDIR%
```

MySQL looks for options in each location first in the `my.ini` file, and then in the `my.cnf` file. However, to avoid confusion, it is best if you use only one file. If your PC uses a boot loader where `C:` is not the boot drive, your only option is to use the `my.ini` file. Whichever option file you use, it must be a plain text file.

Note

When using the MySQL Installer to install MySQL Server, it will create the `my.ini` at the default location. And as of MySQL Server 5.5.27, the user running MySQL Installer is granted full permissions to this new `my.ini`.

In other words, be sure that the MySQL Server user has permission to read the `my.ini` file.

You can also make use of the example option files included with your MySQL distribution; see [Server Configuration Defaults](#).

An option file can be created and modified with any text editor, such as Notepad. For example, if MySQL is installed in `E:\mysql` and the data directory is in `E:\mydata\data`, you can create an option file containing a `[mysqld]` section to specify values for the `basedir` and `datadir` options:

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=E:/mydata/data
```

Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]
# set basedir to your installation path
basedir=E:\\mysql
# set datadir to the location of your data directory
datadir=E:\\mydata\\data
```

The rules for use of backslash in option file values are given in [Using Option Files](#).

The data directory is located within the `AppData` directory for the user running MySQL.

If you would like to use a data directory in a different location, you should copy the entire contents of the `data` directory to the new location. For example, if you want to use `E:\mydata` as the data directory instead, you must do two things:

1. Move the entire `data` directory and all of its contents from the default location (for example `C:\Program Files\MySQL\MySQL Server 5.5\data`) to `E:\mydata`.
2. Use a `--datadir` option to specify the new data directory location each time you start the server.

5.7.3 Selecting a MySQL Server Type

The following table shows the available servers for Windows in MySQL 5.5.

Binary	Description
<code>mysqld</code>	Optimized binary with named-pipe support
<code>mysqld-debug</code>	Like <code>mysqld</code> , but compiled with full debugging and automatic memory allocation checking

All of the preceding binaries are optimized for modern Intel processors, but should work on any Intel i386-class or higher processor.

Each of the servers in a distribution support the same set of storage engines. The `SHOW ENGINES` statement displays which engines a given server supports.

All Windows MySQL 5.5 servers have support for symbolic linking of database directories.

MySQL supports TCP/IP on all Windows platforms. MySQL servers on Windows also support named pipes, if you start the server with the `--enable-named-pipe` option. It is necessary to use this option explicitly because some users have experienced problems with shutting down the MySQL server when named pipes were used. The default is to use TCP/IP regardless of platform because named pipes are slower than TCP/IP in many Windows configurations.

5.7.4 Starting the Server for the First Time

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 5.5`. 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 [Section 5.7.3, “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 5.5\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: '5.5.58' 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 5.5\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 [Section 9.4, “Securing the Initial MySQL Accounts”](#).

5.7.5 Starting MySQL from the Windows Command Line

The MySQL server can be started manually from the command line. This can be done on any version of Windows.

Note

[MySQL Notifier](#) can also be used to start/stop/restart the MySQL server.

To start the `mysqld` server from the command line, you should start a console window (or “DOS window”) and enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqld"
```

The path to `mysqld` may vary depending on the install location of MySQL on your system.

You can stop the MySQL server by executing this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqladmin" -u root shutdown
```

Note

If the MySQL `root` user account has a password, you need to invoke `mysqladmin` with the `-p` option and supply the password when prompted.

This command invokes the MySQL administrative utility `mysqladmin` to connect to the server and tell it to shut down. The command connects as the MySQL `root` user, which is the default administrative account in the MySQL grant system.

Note

Users in the MySQL grant system are wholly independent from any login users under Microsoft Windows.

If `mysqld` doesn't start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. By default, the error log is located in the `C:\Program Files\MySQL\MySQL Server 5.5\data` directory. It is the file with a suffix of `.err`, or may be specified by passing in the `--log-error` option. Alternatively, you can try to start the server with the `--console` option; in this case, the server may display some useful information on the screen that will help solve the problem.

The last option is to start `mysqld` with the `--standalone` and `--debug` options. In this case, `mysqld` writes a log file `C:\mysqld.trace` that should contain the reason why `mysqld` doesn't start. See [The DEBUG Package](#).

Use `mysqld --verbose --help` to display all the options that `mysqld` supports.

5.7.6 Customizing the PATH for MySQL Tools

To make it easier to invoke MySQL programs, you can add the path name of the MySQL `bin` directory to your Windows system `PATH` environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the **Advanced** tab from the **System Properties** menu that appears, and click the **Environment Variables** button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL `bin` directory (for example, `C:\Program Files\MySQL\MySQL Server 5.5\bin`)

Note

There must be a semicolon separating this path from any values present in this field.

Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. You should now be able to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the `mysql` client, and all MySQL command-line utilities such as `mysqladmin` and `mysqldump`.

You should not add the MySQL `bin` directory to your Windows `PATH` if you are running multiple MySQL servers on the same machine.

Warning

You must exercise great care when editing your system `PATH` by hand; accidental deletion or modification of any portion of the existing `PATH` value can leave you with a malfunctioning or even unusable system.

5.7.7 Starting MySQL as a Windows Service

On Windows, the recommended way to run MySQL is to install it as a Windows service, so that MySQL starts and stops automatically when Windows starts and stops. A MySQL server installed as a service can also be controlled from the command line using `NET` commands, or with the graphical `Services` utility. Generally, to install MySQL as a Windows service you should be logged in using an account that has administrator rights.

Note

`MySQL Notifier` can also be used to monitor the status of the MySQL service.

The `Services` utility (the Windows `Service Control Manager`) can be found in the Windows Control Panel (under **Administrative Tools** on Windows 2000, XP, Vista, and Server 2003). To avoid conflicts, it is advisable to close the `Services` utility while performing server installation or removal operations from the command line.

Installing the service

Before installing MySQL as a Windows service, you should first stop the current server if it is running by using the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqladmin"  
-u root shutdown
```

Note

If the MySQL `root` user account has a password, you need to invoke `mysqladmin` with the `-p` option and supply the password when prompted.

This command invokes the MySQL administrative utility `mysqladmin` to connect to the server and tell it to shut down. The command connects as the MySQL `root` user, which is the default administrative account in the MySQL grant system.

Note

Users in the MySQL grant system are wholly independent from any login users under Windows.

Install the server as a service using this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqld" --install
```

The service-installation command does not start the server. Instructions for that are given later in this section.

To make it easier to invoke MySQL programs, you can add the path name of the MySQL `bin` directory to your Windows system `PATH` environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the **Advanced** tab from the **System Properties** menu that appears, and click the **Environment Variables** button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL `bin` directory (for example, `C:\Program Files\MySQL\MySQL Server 5.5\bin`), and there should be a semicolon separating this path from any values present in this field. Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. You should now be able to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the `mysql` client, and all MySQL command-line utilities such as `mysqladmin` and `mysqldump`.

You should not add the MySQL `bin` directory to your Windows `PATH` if you are running multiple MySQL servers on the same machine.

Warning

You must exercise great care when editing your system `PATH` by hand; accidental deletion or modification of any portion of the existing `PATH` value can leave you with a malfunctioning or even unusable system.

The following additional arguments can be used when installing the service:

- You can specify a service name immediately following the `--install` option. The default service name is `MySQL`.
- If a service name is given, it can be followed by a single option. By convention, this should be `--defaults-file=file_name` to specify the name of an option file from which the server should read options when it starts.

The use of a single option other than `--defaults-file` is possible but discouraged. `--defaults-file` is more flexible because it enables you to specify multiple startup options for the server by placing them in the named option file.

- You can also specify a `--local-service` option following the service name. This causes the server to run using the `LocalService` Windows account that has limited system privileges. This account is available only for Windows XP or newer. If both `--defaults-file` and `--local-service` are given following the service name, they can be in any order.

For a MySQL server that is installed as a Windows service, the following rules determine the service name and option files that the server uses:

- If the service-installation command specifies no service name or the default service name (`MySQL`) following the `--install` option, the server uses the a service name of `MySQL` and reads options from the `[mysqld]` group in the standard option files.
- If the service-installation command specifies a service name other than `MySQL` following the `--install` option, the server uses that service name. It reads options from the `[mysqld]` group and the group that has the same name as the service in the standard option files. This enables you to use the `[mysqld]` group for options that should be used by all MySQL services, and an option group with the service name for use by the server installed with that service name.
- If the service-installation command specifies a `--defaults-file` option after the service name, the server reads options the same way as described in the previous item, except that it reads options only from the named file and ignores the standard option files.

As a more complex example, consider the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqld"
      --install MySQL --defaults-file=C:\my-opts.cnf
```

Here, the default service name (`MySQL`) is given after the `--install` option. If no `--defaults-file` option had been given, this command would have the effect of causing the server to read the `[mysqld]` group from the standard option files. However, because the `--defaults-file` option is present, the server reads options from the `[mysqld]` option group, and only from the named file.

Note

On Windows, if the server is started with the `--defaults-file` and `--install` options, `--install` must be first. Otherwise, `mysqld.exe` will attempt to start the MySQL server.

You can also specify options as Start parameters in the Windows `Services` utility before you start the MySQL service.

Finally, before trying to start the MySQL service, make sure the user variables `%TEMP%` and `%TMP%` (and also `%TMPDIR%`, if it has ever been set) for the system user who is to run the service are pointing to a folder to which the user has write access. The default user for running the MySQL service is `LocalSystem`, and the default value for its `%TEMP%` and `%TMP%` is `C:\Windows\Temp`, a directory `LocalSystem` has write

access to by default. However, if there are any changes to that default setup (for example, changes to the user who runs the service or to the mentioned user variables, or the `--tmpdir` option has been used to put the temporary directory somewhere else), the MySQL service might fail to run because write access to the temporary directory has not been granted to the proper user.

Starting the service

Once a MySQL server has been installed as a service, Windows starts the service automatically whenever Windows starts. The service also can be started immediately from the [Services](#) utility, or by using a `NET START MySQL` command. The `NET` command is not case sensitive.

When run as a service, `mysqld` has no access to a console window, so no messages can be seen there. If `mysqld` does not start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. The error log is located in the MySQL data directory (for example, `C:\Program Files\MySQL\MySQL Server 5.5\data`). It is the file with a suffix of `.err`.

When a MySQL server has been installed as a service, and the service is running, Windows stops the service automatically when Windows shuts down. The server also can be stopped manually by using the [Services](#) utility, the `NET STOP MySQL` command, or the `mysqladmin shutdown` command.

You also have the choice of installing the server as a manual service if you do not wish for the service to be started automatically during the boot process. To do this, use the `--install-manual` option rather than the `--install` option:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqld" --install-manual
```

Removing the service

To remove a server that is installed as a service, first stop it if it is running by executing `NET STOP MySQL`. Then use the `--remove` option to remove it:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqld" --remove
```

If `mysqld` is not running as a service, you can start it from the command line. For instructions, see [Section 5.7.5, "Starting MySQL from the Windows Command Line"](#).

If you encounter difficulties during installation, see [Section 5.8, "Troubleshooting a Microsoft Windows MySQL Server Installation"](#).

For more information about stopping or removing a MySQL Windows service, see [Starting Multiple MySQL Instances as Windows Services](#).

5.7.8 Testing The MySQL Installation

You can test whether the MySQL server is working by executing any of the following commands:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqlshow"  
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqlshow" -u root mysql  
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqladmin" version status proc  
C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql" test
```

If `mysqld` is slow to respond to TCP/IP connections from client programs, there is probably a problem with your DNS. In this case, start `mysqld` with the `--skip-name-resolve` option and use only `localhost`

and IP addresses in the `Host` column of the MySQL grant tables. (Be sure that an account exists that specifies an IP address or you may not be able to connect.)

You can force a MySQL client to use a named-pipe connection rather than TCP/IP by specifying the `--pipe` or `--protocol=PIPE` option, or by specifying `.` (period) as the host name. Use the `--socket` option to specify the name of the pipe if you do not want to use the default pipe name.

If you have set a password for the `root` account, deleted the anonymous account, or created a new user account, then to connect to the MySQL server you must use the appropriate `-u` and `-p` options with the commands shown previously. See [Connecting to the MySQL Server](#).

For more information about `mysqlshow`, see [mysqlshow — Display Database, Table, and Column Information](#).

5.8 Troubleshooting a Microsoft Windows MySQL Server Installation

When installing and running MySQL for the first time, you may encounter certain errors that prevent the MySQL server from starting. This section helps you diagnose and correct some of these errors.

Your first resource when troubleshooting server issues is the [error log](#). The MySQL server uses the error log to record information relevant to the error that prevents the server from starting. The error log is located in the [data directory](#) specified in your `my.ini` file. The default data directory location is `C:\Program Files\MySQL\MySQL Server 5.5\data`, or `C:\ProgramData\MySQL` on Windows 7 and Windows Server 2008. The `C:\ProgramData` directory is hidden by default. You need to change your folder options to see the directory and contents. For more information on the error log and understanding the content, see [The Error Log](#).

For information regarding possible errors, also consult the console messages displayed when the MySQL service is starting. Use the `NET START MySQL` command from the command line after installing `mysqld` as a service to see any error messages regarding the starting of the MySQL server as a service. See [Section 5.7.7, “Starting MySQL as a Windows Service”](#).

The following examples show other common error messages you might encounter when installing MySQL and starting the server for the first time:

- If the MySQL server cannot find the `mysql` privileges database or other critical files, it displays these messages:

```
System error 1067 has occurred.
Fatal error: Can't open and lock privilege tables:
Table 'mysql.user' doesn't exist
```

These messages often occur when the MySQL base or data directories are installed in different locations than the default locations (`C:\Program Files\MySQL\MySQL Server 5.5` and `C:\Program Files\MySQL\MySQL Server 5.5\data`, respectively).

This situation can occur when MySQL is upgraded and installed to a new location, but the configuration file is not updated to reflect the new location. In addition, old and new configuration files might conflict. Be sure to delete or rename any old configuration files when upgrading MySQL.

If you have installed MySQL to a directory other than `C:\Program Files\MySQL\MySQL Server 5.5`, ensure that the MySQL server is aware of this through the use of a configuration (`my.ini`) file. Put the `my.ini` file in your Windows directory, typically `C:\WINDOWS`. To determine its exact location from the value of the `WINDIR` environment variable, issue the following command from the command prompt:

```
C:\> echo %WINDIR%
```

You can create or modify an option file with any text editor, such as Notepad. For example, if MySQL is installed in `E:\mysql` and the data directory is `D:\MySQLdata`, you can create the option file and set up a `[mysqld]` section to specify values for the `basedir` and `datadir` options:

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=D:/MySQLdata
```

Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]
# set basedir to your installation path
basedir=C:\\Program Files\\MySQL\\MySQL Server 5.5
# set datadir to the location of your data directory
datadir=D:\\MySQLdata
```

The rules for use of backslash in option file values are given in [Using Option Files](#).

If you change the `datadir` value in your MySQL configuration file, you must move the contents of the existing MySQL data directory before restarting the MySQL server.

See [Section 5.7.2, "Creating an Option File"](#).

- If you reinstall or upgrade MySQL without first stopping and removing the existing MySQL service and install MySQL using the MySQL Installer, you might see this error:

```
Error: Cannot create Windows service for MySql. Error: 0
```

This occurs when the Configuration Wizard tries to install the service and finds an existing service with the same name.

One solution to this problem is to choose a service name other than `mysql` when using the configuration wizard. This enables the new service to be installed correctly, but leaves the outdated service in place. Although this is harmless, it is best to remove old services that are no longer in use.

To permanently remove the old `mysql` service, execute the following command as a user with administrative privileges, on the command line:

```
C:\> sc delete mysql
[SC] DeleteService SUCCESS
```

If the `sc` utility is not available for your version of Windows, download the `delsrv` utility from <http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/delsrv-o.asp> and use the `delsrv mysql` syntax.

5.9 Windows Postinstallation Procedures

GUI tools exist that perform most of the tasks described in this section, including:

- [MySQL Installer](#): Used to install and upgrade MySQL products.
- [MySQL Workbench](#): Manages the MySQL server and edits SQL statements.

- **MySQL Notifier**: Starts, stops, or restarts the MySQL server, and monitors its status.
- **MySQL for Excel**: Edits MySQL data with Microsoft Excel.

On Windows, you need not create the data directory and the grant tables. MySQL Windows distributions include the grant tables with a set of preinitialized accounts in the `mysql` database under the data directory.

Regarding passwords, if you installed MySQL using the MySQL Installer, you may have already assigned passwords to the accounts. (See [Section 5.3, “MySQL Installer for Windows”](#).) Otherwise, use the password-assignment procedure given in [Section 9.4, “Securing the Initial MySQL Accounts”](#).

Before assigning passwords, you might want to try running some client programs to make sure that you can connect to the server and that it is operating properly. Make sure that the server is running (see [Section 5.7.4, “Starting the Server for the First Time”](#)). You can also set up a MySQL service that runs automatically when Windows starts (see [Section 5.7.7, “Starting MySQL as a Windows Service”](#)).

These instructions assume that your current location is the MySQL installation directory and that it has a `bin` subdirectory containing the MySQL programs used here. If that is not true, adjust the command path names accordingly.

If you installed MySQL using MySQL Installer (see [Section 5.3, “MySQL Installer for Windows”](#)), the default installation directory is `C:\Program Files\MySQL\MySQL Server 5.5`:

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 5.5"
```

A common installation location for installation from a Zip package is `C:\mysql`:

```
C:\> cd C:\mysql
```

Alternatively, add the `bin` directory to your `PATH` environment variable setting. That enables your command interpreter to find MySQL programs properly, so that you can run a program by typing only its name, not its path name. See [Section 5.7.6, “Customizing the PATH for MySQL Tools”](#).

With the server running, issue the following commands to verify that you can retrieve information from the server. The output should be similar to that shown here.

Use `mysqlshow` to see what databases exist:

```
C:\> bin\mysqlshow
+-----+
|   Databases   |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| test          |
+-----+
```

The list of installed databases may vary, but will always include the minimum of `mysql` and `information_schema`.

The preceding command (and commands for other MySQL programs such as `mysql`) may not work if the correct MySQL account does not exist. For example, the program may fail with an error, or you may not be able to view all databases. If you installed MySQL using MySQL Installer, the `root` user will have been created automatically with the password you supplied. In this case, you should use the `-u root` and `-p` options. (You must use those options if you have already secured the initial MySQL accounts.) With `-p`, the client program prompts for the `root` password. For example:

```
C:\> bin\mysqlshow -u root -p
Enter password: (enter root password here)
+-----+
|      Databases      |
+-----+
| information_schema  |
| mysql               |
| performance_schema  |
| test                |
+-----+
```

If you specify a database name, `mysqlshow` displays a list of the tables within the database:

```
C:\> bin\mysqlshow mysql
Database: mysql
+-----+
|      Tables        |
+-----+
| columns_priv       |
| db                  |
| event              |
| func                |
| general_log         |
| help_category      |
| help_keyword        |
| help_relation       |
| help_topic          |
| host                |
| ndb_binlog_index    |
| plugin              |
| proc                |
| procs_priv          |
| proxies_priv        |
| servers              |
| slow_log            |
| tables_priv         |
| time_zone           |
| time_zone_leap_second |
| time_zone_name      |
| time_zone_transition |
| time_zone_transition_type |
| user                |
+-----+
```

Use the `mysql` program to select information from a table in the `mysql` database:

```
C:\> bin\mysql -e "SELECT User, Host, plugin FROM mysql.user" mysql
+-----+
| User | Host      | plugin                |
+-----+
| root | localhost | mysql_native_password |
+-----+
```

For more information about `mysql` and `mysqlshow`, see [mysql — The MySQL Command-Line Tool](#), and [mysqlshow — Display Database, Table, and Column Information](#).

5.10 Upgrading MySQL on Windows

To upgrade MySQL on Windows, follow these steps:

1. Review [Section 10.1, “Upgrading MySQL”](#), for additional information on upgrading MySQL that is not specific to Windows.

2. Always back up your current MySQL installation before performing an upgrade. See [Database Backup Methods](#).
3. Download the latest Windows distribution of MySQL from <http://dev.mysql.com/downloads/>.
4. Before upgrading MySQL, stop the server. If the server is installed as a service, stop the service with the following command from the command prompt:

```
C:\> NET STOP MySQL
```

If you are not running the MySQL server as a service, use `mysqladmin` to stop it. For example, before upgrading from MySQL 5.1 to 5.5, use `mysqladmin` from MySQL 5.1 as follows:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqladmin" -u root shutdown
```

Note

If the MySQL `root` user account has a password, invoke `mysqladmin` with the `-p` option and enter the password when prompted.

5. Before upgrading to MySQL 5.5 from a version previous to 4.1.5, or from a version of MySQL installed from a Zip archive to a version of MySQL installed with the MySQL Installation Wizard, you must first manually remove the previous installation and MySQL service (if the server is installed as a service).

To remove the MySQL service, use the following command:

```
C:\> C:\mysql\bin\mysqld --remove
```

If you do not remove the existing service, the MySQL Installation Wizard may fail to properly install the new MySQL service.

6. If you are using the MySQL Installer, start it as described in [Section 5.3, “MySQL Installer for Windows”](#).

If you are using the MySQL Installation Wizard, start the wizard as described in [Section 5.5.1, “Using the MySQL Installation Wizard”](#).

7. If you are upgrading MySQL from a Zip archive, extract the archive. You may either overwrite your existing MySQL installation (usually located at `C:\mysql`), or install it into a different directory, such as `C:\mysql5`. Overwriting the existing installation is recommended. However, for upgrades (as opposed to installing for the first time), you must remove the data directory from your existing MySQL installation to avoid replacing your current data files. To do so, follow these steps:
 - a. Unzip the Zip archive in some location other than your current MySQL installation
 - b. Remove the data directory
 - c. Rezip the Zip archive
 - d. Unzip the modified Zip archive on top of your existing installation

Alternatively:

- a. Unzip the Zip archive in some location other than your current MySQL installation
- b. Remove the data directory

- c. Move the data directory from the current MySQL installation to the location of the just-removed data directory
 - d. Remove the current MySQL installation
 - e. Move the unzipped installation to the location of the just-removed installation
8. If you were running MySQL as a Windows service and you had to remove the service earlier in this procedure, reinstall the service. (See [Section 5.7.7, “Starting MySQL as a Windows Service”](#).)
 9. Restart the server. For example, use `NET START MySQL` if you run MySQL as a service, or invoke `mysqld` directly otherwise.
 10. As Administrator, run `mysql_upgrade` to check your tables, attempt to repair them if necessary, and update your grant tables if they have changed so that you can take advantage of any new capabilities. See [mysql_upgrade — Check and Upgrade MySQL Tables](#).
 11. If you encounter errors, see [Section 5.8, “Troubleshooting a Microsoft Windows MySQL Server Installation”](#).

Chapter 6 Installing MySQL on OS X

Table of Contents

6.1 General Notes on Installing MySQL on OS X	123
6.2 Installing MySQL on OS X Using Native Packages	124
6.3 Installing a MySQL Launch Daemon	130
6.4 Installing and Using the MySQL Preference Pane	133

For a list of OS X versions that the MySQL server supports, see <http://www.mysql.com/support/supportedplatforms/database.html>.

MySQL for OS X is available in a number of different forms:

- Native Package Installer, which uses the native OS X installer (DMG) to walk you through the installation of MySQL. For more information, see [Section 6.2, “Installing MySQL on OS X Using Native Packages”](#). You can use the package installer with OS X. The user you use to perform the installation must have administrator privileges.
- Compressed TAR archive, which uses a file packaged using the Unix `tar` and `gzip` commands. To use this method, you will need to open a `Terminal` window. You do not need administrator privileges using this method, as you can install the MySQL server anywhere using this method. For more information on using this method, you can use the generic instructions for using a tarball, [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#).

In addition to the core installation, the Package Installer also includes [Section 6.3, “Installing a MySQL Launch Daemon”](#) and [Section 6.4, “Installing and Using the MySQL Preference Pane”](#), both of which simplify the management of your installation.

For additional information on using MySQL on OS X, see [Section 6.1, “General Notes on Installing MySQL on OS X”](#).

6.1 General Notes on Installing MySQL on OS X

You should keep the following issues and notes in mind:

- As of MySQL server 5.5.45, the DMG bundles a `launchd` daemon instead of the deprecated startup item. Startup items do not function as of OS X 10.10 (Yosemite), so using `launchd` is preferred. The available MySQL preference pane under OS X **System Preferences** was also updated to use `launchd`.
- You may need (or want) to create a specific `mysql` user to own the MySQL directory and data. You can do this through the `Directory Utility`, and the `mysql` user should already exist. For use in single user mode, an entry for `_mysql` (note the underscore prefix) should already exist within the `system / etc/passwd` file.
- Because the MySQL package installer installs the MySQL contents into a version and platform specific directory, you can use this to upgrade and migrate your database between versions. You will need to either copy the `data` directory from the old version to the new version, or alternatively specify an alternative `datadir` value to set location of the data directory. By default, the MySQL directories are installed under `/usr/local/`.

- You might want to add aliases to your shell's resource file to make it easier to access commonly used programs such as `mysql` and `mysqladmin` from the command line. The syntax for `bash` is:

```
alias mysql=/usr/local/mysql/bin/mysql
alias mysqladmin=/usr/local/mysql/bin/mysqladmin
```

For `tcsh`, use:

```
alias mysql /usr/local/mysql/bin/mysql
alias mysqladmin /usr/local/mysql/bin/mysqladmin
```

Even better, add `/usr/local/mysql/bin` to your `PATH` environment variable. You can do this by modifying the appropriate startup file for your shell. For more information, see [Invoking MySQL Programs](#).

- After you have copied over the MySQL database files from the previous installation and have successfully started the new server, you should consider removing the old installation files to save disk space. Additionally, you should also remove older versions of the Package Receipt directories located in `/Library/Receipts/mysql-VERSION.pkg`.
- Prior to OS X 10.7, MySQL server was bundled with OS X Server.

6.2 Installing MySQL on OS X Using Native Packages

The package is located inside a disk image (`.dmg`) file that you first need to mount by double-clicking its icon in the Finder. It should then mount the image and display its contents.

Note

Before proceeding with the installation, be sure to stop all running MySQL server instances by using either the MySQL Manager Application (on OS X Server), the preference pane, or `mysqladmin shutdown` on the command line.

When installing from the package version, you can also install the MySQL preference pane, which will enable you to control the startup and execution of your MySQL server from System Preferences. For more information, see [Section 6.4, "Installing and Using the MySQL Preference Pane"](#).

When installing using the package installer, the files are installed into a directory within `/usr/local` matching the name of the installation version and platform. For example, the installer file `mysql-5.5.58-osx10.9-x86_64.dmg` installs MySQL into `/usr/local/mysql-5.5.58-osx10.9-x86_64/`. The following table shows the layout of the installation directory.

Table 6.1 MySQL Installation Layout on OS X

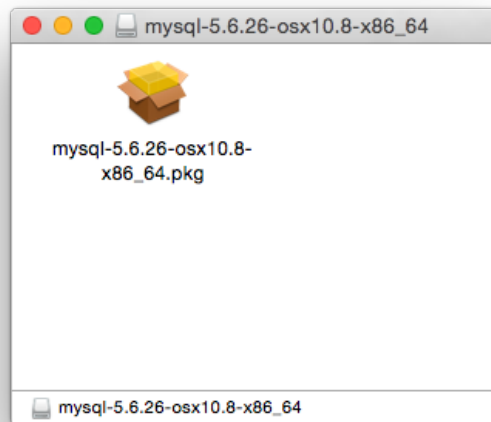
Directory	Contents of Directory
<code>bin, scripts</code>	<code>mysqld</code> server, client and utility programs
<code>data</code>	Log files, databases
<code>docs</code>	Helper documents, like the Release Notes and build information
<code>include</code>	Include (header) files
<code>lib</code>	Libraries
<code>man</code>	Unix manual pages

Directory	Contents of Directory
<code>mysql-test</code>	MySQL test suite
<code>share</code>	Miscellaneous support files, including error messages, sample configuration files, SQL for database installation
<code>sql-bench</code>	Benchmarks
<code>support-files</code>	Scripts and sample configuration files
<code>/tmp/mysql.sock</code>	Location of the MySQL Unix socket

During the package installer process, a symbolic link from `/usr/local/mysql` to the version/platform specific directory created during installation will be created automatically.

1. Download and open the MySQL package installer, which is provided on a disk image (`.dmg`) that includes the main MySQL installation package file. Double-click the disk image to open it.

Figure 6.1 MySQL Package Installer: DMG Contents



2. Double-click the MySQL installer package. It will be named according to the version of MySQL you have downloaded. For example, if you have downloaded MySQL server 5.5.58, double-click `mysql-5.5.58-osx-10.9-x86_64.pkg`.
3. You will be presented with the opening installer dialog. Click **Continue** to begin installation.

Figure 6.2 MySQL Package Installer: Introduction



4. If you have downloaded the community version of MySQL, you will be shown a copy of the relevant GNU General Public License. Click **Continue** and then **Agree** to continue.
5. From the **Installation Type** page you can either click **Install** to execute the installation wizard using all defaults, click **Customize** to alter which components to install (MySQL server, Preference Pane, Launchd Support -- all enabled by default), or click **Change Installation Location** to change the type of installation for either all users, only the user executing the Installer, or define a custom location.

Figure 6.3 MySQL Package Installer: Installation Type

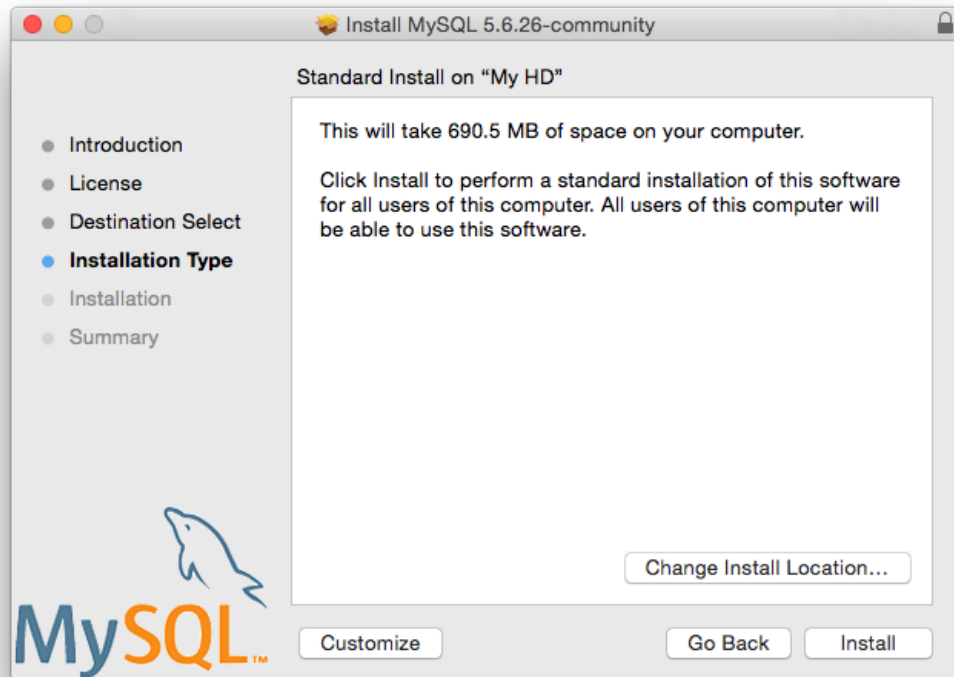


Figure 6.4 MySQL Package Installer: Destination Select (Change Installation Location)

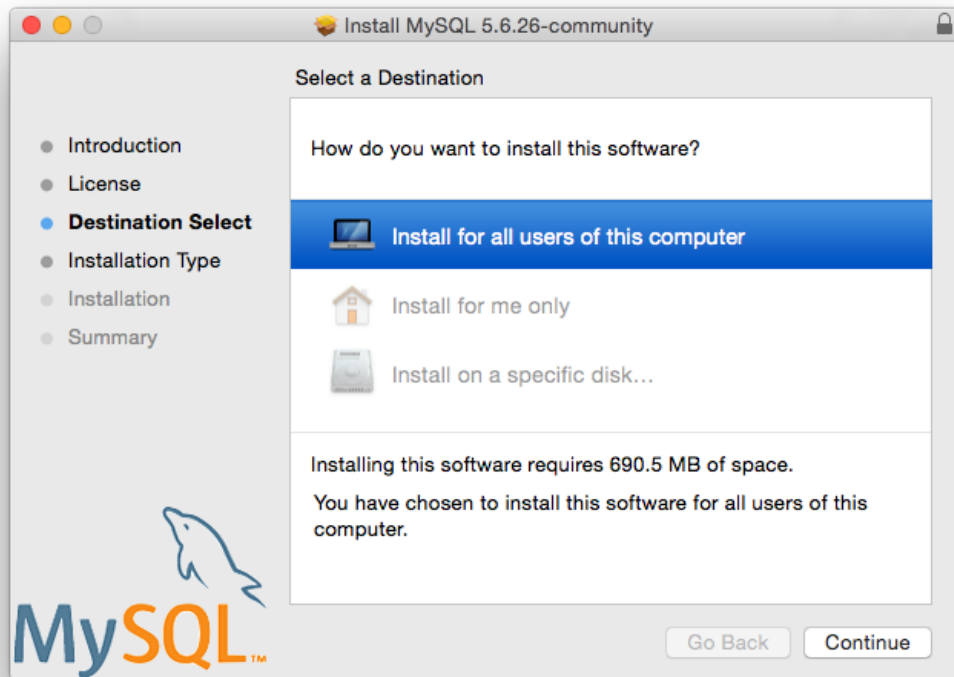
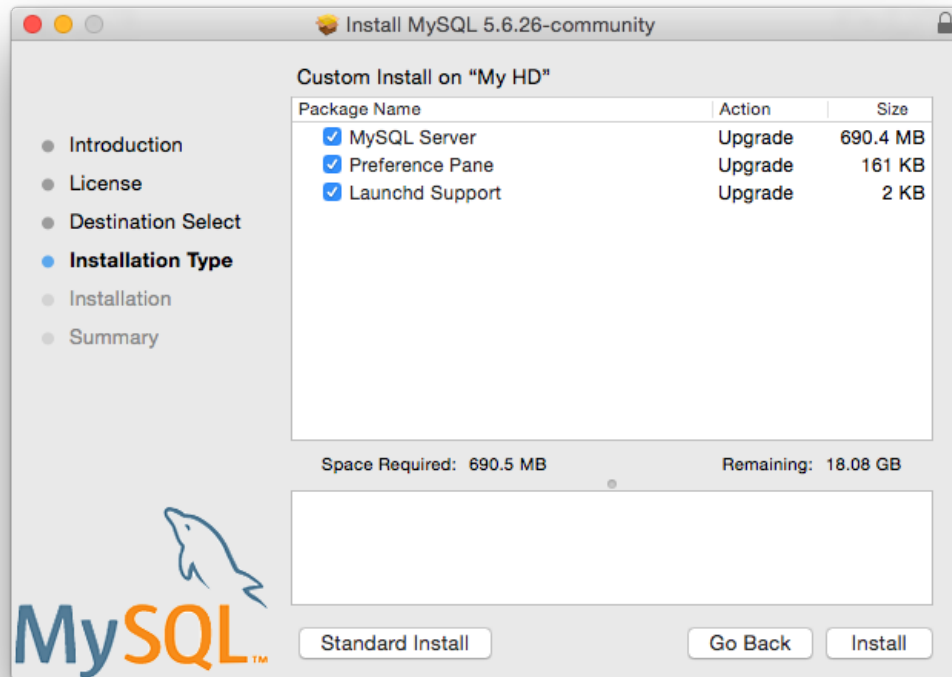


Figure 6.5 MySQL Package Installer: Customize



6. Click **Install** to begin the installation process.
7. Once the installation has been completed successfully, you will be shown an **Install Succeeded** message with a short summary. Now, **Close** the wizard and begin using the MySQL server.

Figure 6.6 MySQL Package Installer: Summary



MySQL server is now installed, but it is not loaded (started) by default. Use either `launchctl` from the command line, or start MySQL by clicking "Start" using the MySQL preference pane. For additional information, see [Section 6.3, "Installing a MySQL Launch Daemon"](#), and [Section 6.4, "Installing and Using the MySQL Preference Pane"](#).

6.3 Installing a MySQL Launch Daemon

OS X uses launch daemons to automatically start, stop, and manage processes and applications such as MySQL.

Note

Before MySQL 5.5.45, the OS X builds installed startup items instead of launchd daemons. However, startup items do not function as of OS X 10.10 (Yosemite). The OS X builds now install launchd daemons.

By default, the installation package (DMG) on OS X installs a launchd file named `/Library/LaunchDaemons/com.oracle.oss.mysql.mysqld.plist` that contains a plist definition similar to:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
  "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
```



```
<dict>
  <key>Label</key>          <string>com.oracle.oss.mysql.mysql</string>
  <key>ProcessType</key>    <string>Interactive</string>
  <key>Disabled</key>       <false/>
  <key>RunAtLoad</key>      <true/>
  <key>KeepAlive</key>      <true/>
  <key>SessionCreate</key>  <true/>
  <key>LaunchOnlyOnce</key> <false/>
  <key>UserName</key>       <string>_mysql</string>
  <key>GroupName</key>     <string>_mysql</string>
  <key>ExitTimeOut</key>    <integer>600</integer>
  <key>Program</key>       <string>/usr/local/mysql/bin/mysqld</string>
  <key>ProgramArguments</key>
    <array>
      <string>/usr/local/mysql/bin/mysqld</string>
      <string>--user=_mysql</string>
      <string>--basedir=/usr/local/mysql</string>
      <string>--datadir=/usr/local/mysql/data</string>
      <string>--plugin-dir=/usr/local/mysql/lib/plugin</string>
      <string>--log-error=/usr/local/mysql/data/mysqld.local.err</string>
      <string>--pid-file=/usr/local/mysql/data/mysqld.local.pid</string>
      <string>--port=3306</string>
    </array>
  <key>WorkingDirectory</key> <string>/usr/local/mysql</string>
</dict>
</plist>
```

Note

Some users report that adding a plist DOCTYPE declaration causes the launchd operation to fail, despite it passing the lint check. We suspect it's a copy-n-paste error. The md5 checksum of a file containing the above snippet is 60d7963a0bb2994b69b8b9c123db09df.

To enable the launchd service, you can either:

- Click **Start MySQL Server** from the MySQL preference pane.

Figure 6.7 MySQL Preference Pane: Location

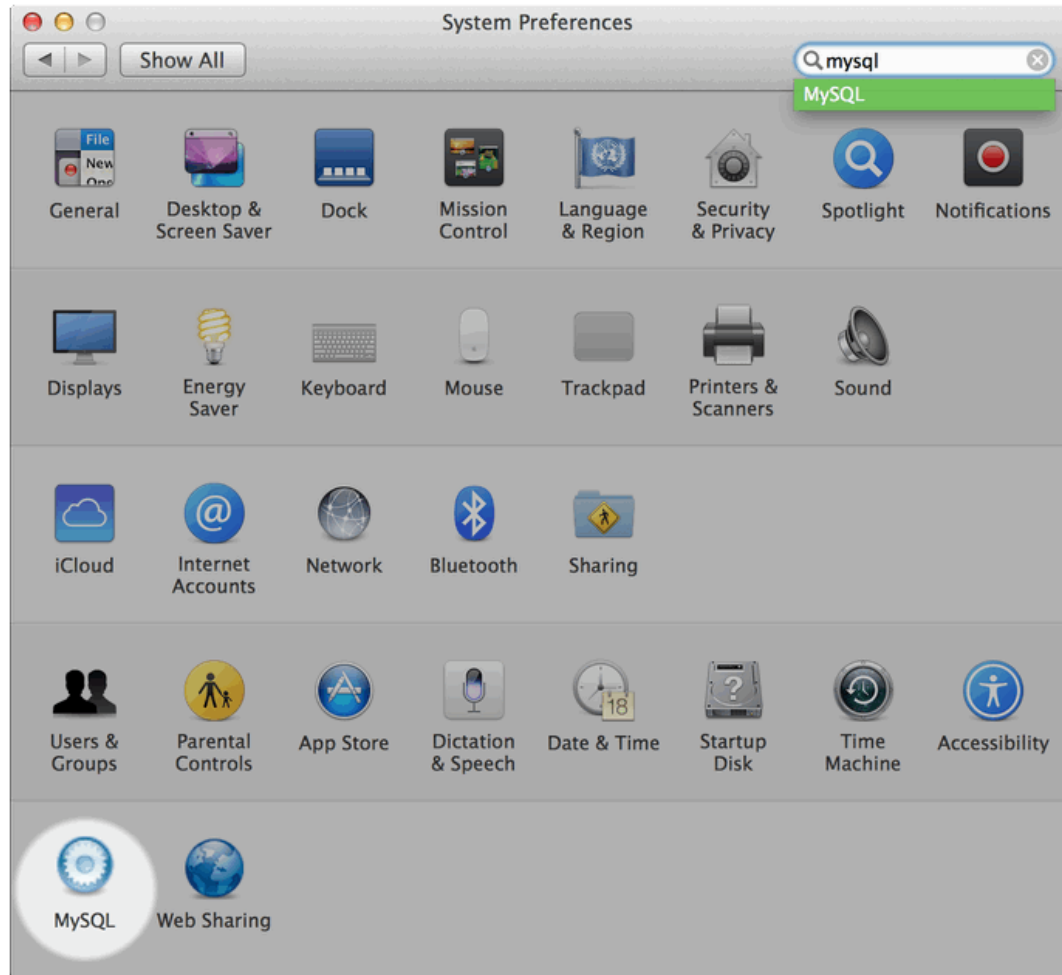
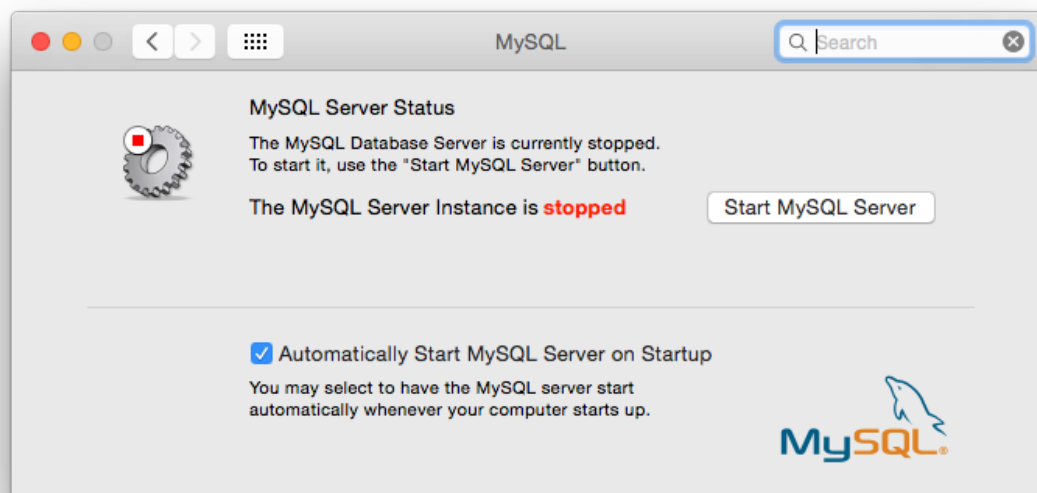


Figure 6.8 MySQL Preference Pane: Usage

- Or, manually load the launchd file.

```
shell> cd /Library/LaunchDaemons
shell> sudo launchctl load -F com.oracle.oss.mysql.mysql.plist
```

Note

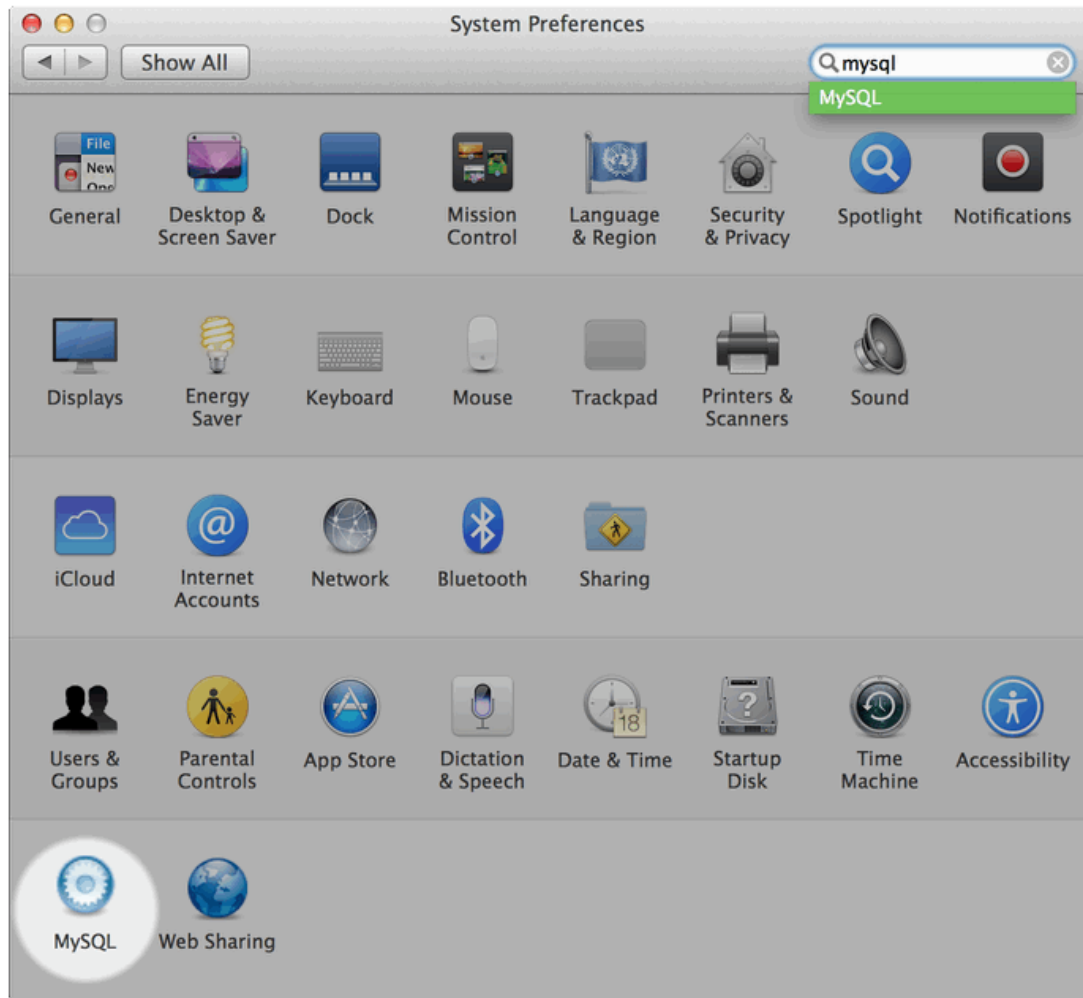
When upgrading MySQL server, the launchd installation process will remove the old startup items that were installed with MySQL server 5.5.44 and below.

6.4 Installing and Using the MySQL Preference Pane

The MySQL Installation Package includes a MySQL preference pane that enables you to start, stop, and control automated startup during boot of your MySQL installation.

This preference pane is installed by default, and is listed under your system's *System Preferences* window.

Figure 6.9 MySQL Preference Pane: Location



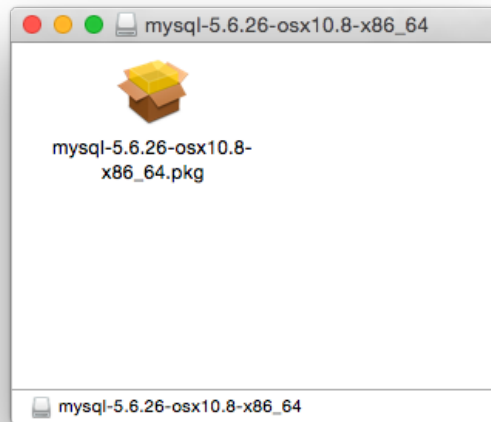
To install the MySQL Preference Pane:

1. Download and open the MySQL package installer, which is provided on a disk image (.dmg) that includes the main MySQL installation package.

Note

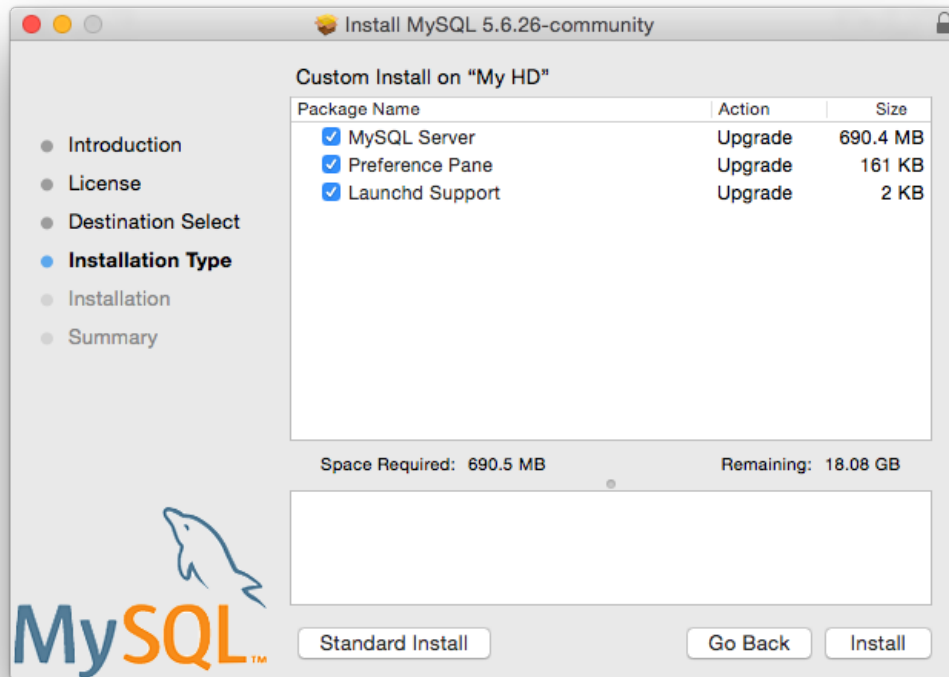
Before MySQL 5.5.45, OS X packages included the deprecated startup items instead of launchd daemons, and the preference pane managed that instead of launchd.

Figure 6.10 MySQL Package Installer: DMG Contents



2. Go through the process of installing the MySQL server, as described in the documentation at [Section 6.2, "Installing MySQL on OS X Using Native Packages"](#).
3. Click **Customize** at the **Installation Type** step. The "Preference Pane" option is listed there and enabled by default.

Figure 6.11 MySQL Installer on OS X: Customize



4. Complete the MySQL server installation process.

Note

The MySQL preference pane only starts and stops MySQL installation installed from the MySQL package installation that have been installed in the default location.

Once the MySQL preference pane has been installed, you can control your MySQL server instance using the preference pane. To use the preference pane, open the **System Preferences...** from the Apple menu. Select the MySQL preference pane by clicking the MySQL logo within the **bottom** section of the preference panes list.

Figure 6.12 MySQL Preference Pane: Location

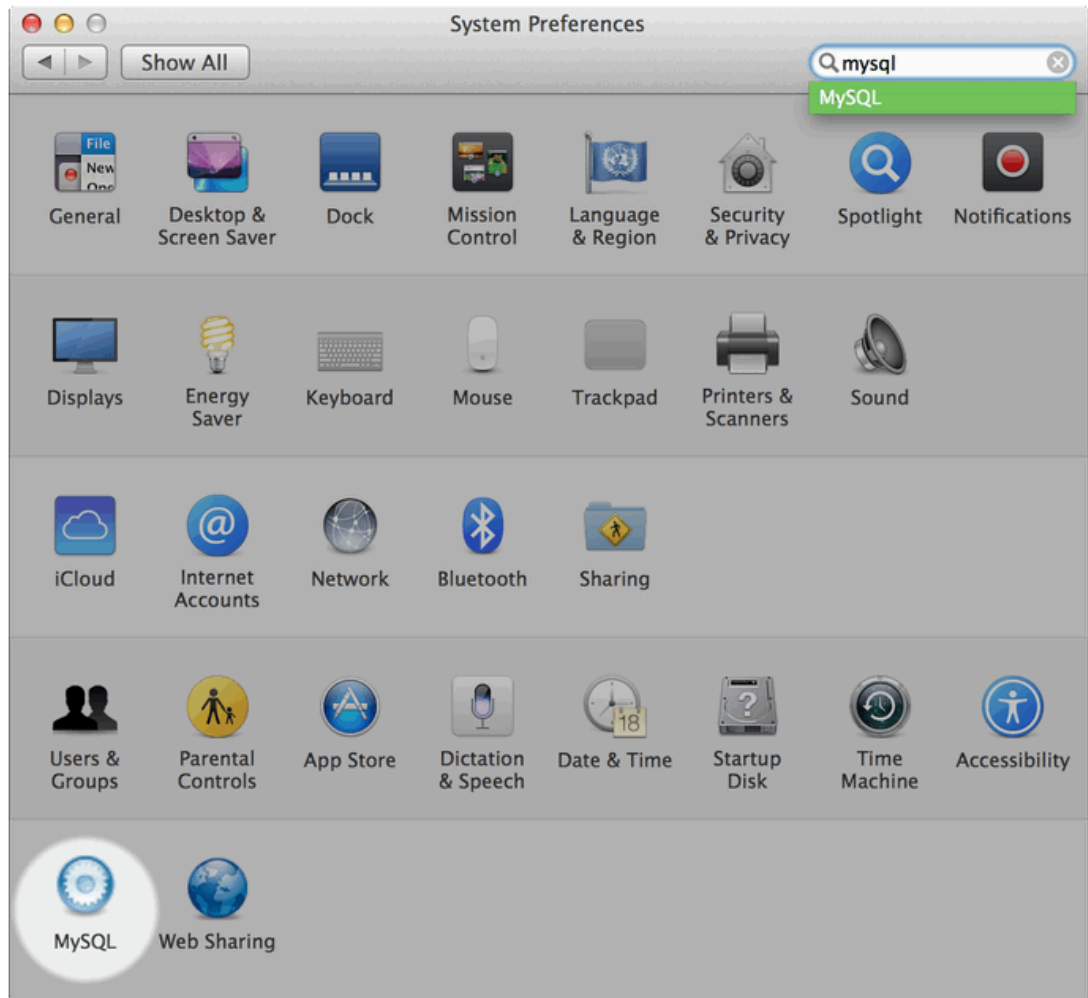


Figure 6.13 MySQL Preference Pane: Usage

The MySQL Preference Pane shows the current status of the MySQL server, showing **stopped** (in red) if the server is not running and **running** (in green) if the server has already been started. The preference pane also shows the current setting for whether the MySQL server has been set to start automatically.

- **To start the MySQL server using the preference pane:**

Click **Start MySQL Server**. You may be prompted for the username and password of a user with administrator privileges to start the MySQL server.

- **To stop the MySQL server using the preference pane:**

Click **Stop MySQL Server**. You may be prompted for the username and password of a user with administrator privileges to stop the MySQL server.

- **To automatically start the MySQL server when the system boots:**

Check the check box next to **Automatically Start MySQL Server on Startup**.

- **To disable automatic MySQL server startup when the system boots:**

Uncheck the check box next to **Automatically Start MySQL Server on Startup**.

You can close the [System Preferences...](#) window once you have completed your settings.

Chapter 7 Installing MySQL on Linux

Table of Contents

7.1 Installing MySQL on Linux Using RPM Packages	139
7.2 Installing MySQL on Linux Using Debian Packages	144
7.3 Installing MySQL on Linux Using Native Package Managers	145

Linux supports a number of different solutions for installing MySQL. The recommended method is to use one of the distributions from Oracle. If you choose this method, there are several options available:

- Installing from a generic binary package in `.tar.gz` format. See [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#) for more information.
- Extracting and compiling MySQL from a source distribution. For detailed instructions, see [Chapter 4, *Installing MySQL from Source*](#).
- Installing using a precompiled RPM package. For more information, see [Section 7.1, “Installing MySQL on Linux Using RPM Packages”](#).
- Installing using a precompiled Debian package. For more information, see [Section 7.2, “Installing MySQL on Linux Using Debian Packages”](#).
- Installing using Oracle's Unbreakable Linux Network (ULN). For more information, see [Installing MySQL Using Unbreakable Linux Network \(ULN\)](#).

As an alternative, you can use the native package manager within your Linux distribution to automatically download and install MySQL for you. Native package installations can take care of the download and dependencies required to run MySQL, but the MySQL version will often be some versions behind the currently available release. You will also normally be unable to install development releases, as these are not usually made available in the native repository. For more information on using the native package installers, see [Section 7.3, “Installing MySQL on Linux Using Native Package Managers”](#).

Note

For many Linux installations, you will want to set up MySQL to be started automatically when your machine starts. Many of the native package installations perform this operation for you, but for source, binary and RPM solutions you may need to set this up separately. The required script, `mysql.server`, can be found in the `support-files` directory under the MySQL installation directory or in a MySQL source tree. You can install it as `/etc/init.d/mysql` for automatic MySQL startup and shutdown. See [mysql.server — MySQL Server Startup Script](#).

7.1 Installing MySQL on Linux Using RPM Packages

Note

To install or upgrade to MySQL 5.5.31, be sure to read the special instructions at the end of this section.

The recommended way to install MySQL on RPM-based Linux distributions is by using the RPM packages. The RPMs that we provide to the community should work on all versions of Linux that support RPM packages and use `glibc` 2.3. To obtain RPM packages, see [Section 2.2, “How to Get MySQL”](#).

For non-RPM Linux distributions, you can install MySQL using a `.tar.gz` package. See [Chapter 3, Installing MySQL on Unix/Linux Using Generic Binaries](#).

Installations created from our Linux RPM distributions result in files under the system directories shown in the following table.

Table 7.1 MySQL Installation Layout for Linux RPM Packages

Directory	Contents of Directory
<code>/usr/bin</code>	Client programs and scripts
<code>/usr/sbin</code>	The <code>mysqld</code> server
<code>/var/lib/mysql</code>	Log files, databases
<code>/var/lib/mysql-files</code>	Value of <code>secure_file_priv</code>
<code>/usr/share/info</code>	MySQL manual in Info format
<code>/usr/share/man</code>	Unix manual pages
<code>/usr/include/mysql</code>	Include (header) files
<code>/usr/lib/mysql</code>	Libraries
<code>/usr/share/mysql</code>	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation
<code>/usr/share/sql-bench</code>	Benchmarks

Note

RPM distributions of MySQL are also provided by other vendors. Be aware that they may differ from those built by Oracle in features, capabilities, and conventions (including communication setup), and that the instructions in this manual do not necessarily apply to installing them. The vendor's instructions should be consulted instead. Because of these differences, RPM packages built by Oracle check whether such RPMs built by other vendors are installed. If so, the RPM does not install and produces a message explaining this.

Conflicts can arise when an RPM from another vendor is already installed, such as when a vendor's conventions about which files belong with the server and which belong with the client library differ from the breakdown used for Oracle packages. In such cases, attempts to install an Oracle RPM with `rpm -i` may result in messages that files in the RPM to be installed conflict with files from an installed package (denoted `mysql-libs` in the following paragraphs).

Each MySQL release provides a `MySQL-shared-compat` package that is meant to replace `mysql-libs` and provides a replacement-compatible client library for older MySQL series. `MySQL-shared-compat` is set up to make `mysql-libs` obsolete, but `rpm` explicitly refuses to replace obsoleted packages when invoked with `-i` (unlike `-U`), which is why installation with `rpm -i` produces a conflict.

`MySQL-shared-compat` can safely be installed alongside `mysql-libs` because libraries are installed to different locations. Therefore, it is possible to install `MySQL-shared-compat` first, then manually remove `mysql-libs` before continuing with the installation. After `mysql-libs` is removed, the dynamic linker stops looking for the client library in the location where `mysql-libs` puts it, and the library provided by the `MySQL-shared-compat` package takes over.

Another alternative is to install packages using `yum`. In a directory containing all RPM packages for a MySQL release, `yum install MySQL*rpm` installs them in the correct order and removes `mysql-libs` in one step without conflicts.

In most cases, you need install only the `MySQL-server` and `MySQL-client` packages to get a functional standard MySQL installation. The other packages are not required for a standard installation.

RPMs for NDB Cluster. Standard MySQL server RPMs built by MySQL do not provide support for the `NDBCLUSTER` storage engine.

Important

When upgrading an NDB Cluster RPM installation, you must upgrade *all* installed RPMs, including the `Server` and `Client` RPMs.

For more information about installing NDB Cluster from RPMs, see [NDB Cluster Installation and Upgrades](#).

For upgrades, if your installation was originally produced by installing multiple RPM packages, it is best to upgrade all the installed packages, not just some. For example, if you previously installed the server and client RPMs, do not upgrade just the server RPM.

If the data directory exists at RPM installation time, the installation process does not modify existing data. This has the effect, for example, that accounts in the grant tables are not initialized to the default set of accounts.

If you get a dependency failure when trying to install MySQL packages (for example, `error: removing these packages would break dependencies: libmysqlclient.so.10 is needed by ...`), you should also install the `MySQL-shared-compat` package, which includes the shared libraries for older releases for backward compatibility.

The following list shows the available RPM packages. The names shown here use a suffix of `.glibc23.i386.rpm`, but particular packages can have different suffixes, described later. If you plan to install multiple RPM packages, you may wish to download the RPM Bundle `tar` file instead, which contains multiple RPM packages so that you need not download them separately.

- `MySQL-server-VERSION.glibc23.i386.rpm`

The MySQL server. You need this unless you only want to connect to a MySQL server running on another machine.

- `MySQL-client-VERSION.glibc23.i386.rpm`

The standard MySQL client programs. You probably always want to install this package.

- `MySQL-devel-VERSION.glibc23.i386.rpm`

The libraries and include files needed to compile other MySQL clients, such as the Perl MySQL module. Install this RPM if you intend to compile C API applications.

- `MySQL-shared-VERSION.glibc23.i386.rpm`

The shared libraries (`libmysqlclient.so*`) that certain languages and applications need to dynamically load and use MySQL. It contains single-threaded and thread-safe libraries. Install this RPM if you intend to compile or run C API applications that depend on the shared client library. Prior to MySQL 5.5.6, if you install this package, do not install the `MySQL-shared-compat` package.

- `MySQL-shared-compat-VERSION.glibc23.i386.rpm`

The shared libraries for older releases. It contains single-threaded and thread-safe libraries. Install this package if you have applications installed that are dynamically linked against older versions of MySQL but you want to upgrade to the current version without breaking the library dependencies. Before MySQL 5.5.6, `MySQL-shared-compat` also includes the libraries for the current release, so if you install it, you should not also install `MySQL-shared`. As of 5.5.6, `MySQL-shared-compat` does not include the current library version, so there is no conflict.

As of MySQL 5.5.23, the `MySQL-shared-compat` RPM package enables users of Red Hat-provided `mysql-*-5.1` RPM packages to migrate to Oracle-provided `MySQL-*-5.5` packages. `MySQL-shared-compat` replaces the Red Hat `mysql-libs` package by replacing `libmysqlclient.so` files of the latter package, thus satisfying dependencies of other packages on `mysql-libs`. This change affects only users of Red Hat (or Red Hat-compatible) RPM packages. Nothing is different for users of Oracle RPM packages.

- `MySQL-embedded-VERSION.glibc23.i386.rpm`

The embedded MySQL server library.

- `MySQL-test-VERSION.glibc23.i386.rpm`

The MySQL test suite.

- `MySQL-VERSION.src.rpm`

The source code for all of the previous packages. It can also be used to rebuild the RPMs on other architectures (for example, SPARC).

In RPM package names, the suffix (following the `VERSION` value) has the following syntax:

```
.PLATFORM.CPU.rpm
```

The `PLATFORM` and `CPU` values indicate the type of system for which the package is built. `PLATFORM` indicates the platform and `CPU` indicates the processor type or family.

All packages are dynamically linked against `glibc` 2.3. The `PLATFORM` value indicates whether the package is platform independent or intended for a specific platform, as shown in the following table.

Table 7.2 MySQL Linux RPM Package Platforms

<code>PLATFORM</code> Value	Intended Use
<code>glibc23</code>	Platform independent, should run on any Linux distribution that supports <code>glibc</code> 2.3
<code>rhel4, rhel5</code>	Red Hat Enterprise Linux 4 or 5
<code>el6</code>	Enterprise Linux 6
<code>sles10, sles11</code>	SuSE Linux Enterprise Server 10 or 11

In MySQL 5.5, only `glibc23` packages are available currently.

The `CPU` value indicates the processor type or family for which the package is built, as shown in the following table.

Table 7.3 MySQL Linux RPM Package CPU Identifiers

<code>CPU</code> Value	Intended Processor Type or Family
<code>i386, i586, i686</code>	Pentium processor or better, 32 bit

CPU Value	Intended Processor Type or Family
x86_64	64-bit x86 processor
ia64	Itanium (IA-64) processor

To see all files in an RPM package (for example, a `MySQL-server` RPM), run a command like this (modify the platform and CPU identifiers appropriately for your system):

```
shell> rpm -qpl MySQL-server-VERSION.glibc23.i386.rpm
```

To perform a standard minimal installation, install the server and client RPMs:

```
shell> rpm -i MySQL-server-VERSION.glibc23.i386.rpm
shell> rpm -i MySQL-client-VERSION.glibc23.i386.rpm
```

To install only the client programs, install just the client RPM:

```
shell> rpm -i MySQL-client-VERSION.glibc23.i386.rpm
```

RPM provides a feature to verify the integrity and authenticity of packages before installing them. To learn more about this feature, see [Section 2.3, “Verifying Package Integrity Using MD5 Checksums or GnuPG”](#).

The server RPM places data under the `/var/lib/mysql` directory. The RPM also creates a login account for a user named `mysql` (if one does not exist) to use for running the MySQL server, and creates the appropriate entries in `/etc/init.d/` to start the server automatically at boot time. (This means that if you have performed a previous installation and have made changes to its startup script, you may want to make a copy of the script so that you can reinstall it after you install a newer RPM.) See [Section 9.5, “Starting and Stopping MySQL Automatically”](#), for more information on how MySQL can be started automatically at system startup.

In MySQL 5.5.5 and later, during a new installation using RPM packages, the server boot scripts are installed, but the MySQL server is not started at the end of the installation, since the status of the server during an unattended installation is not known.

In MySQL 5.5.5 and later, during an upgrade installation using RPM packages, if the MySQL server is running when the upgrade occurs, the MySQL server is stopped, the upgrade occurs, and the MySQL server is restarted. If the MySQL server is not already running when the RPM upgrade occurs, the MySQL server is not started at the end of the installation.

If something goes wrong, you can find more information in the binary installation section. See [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#).

Note

The accounts created in the MySQL grant tables for an RPM installation initially have no passwords. After starting the server, you should assign passwords to them using the instructions in [Chapter 9, *Postinstallation Setup and Testing*](#).

An RPM installation creates a user named `mysql` and a group named `mysql` on the system using the `useradd`, `groupadd`, and `usermod` commands. Those commands require appropriate administrative privileges, which is required for locally managed users and groups (as listed in the `/etc/passwd` and `/etc/group` files) by the RPM installation process being run by `root`.

If you log in as the `mysql` user, you may find that MySQL displays “Invalid (old?) table or database name” errors that mention `.mysqlgui`, `lost+found`, `.mysqlgui`, `.bash_history`, `.fonts.cache-1`, `.lesshtst`, `.mysql_history`, `.profile`, `.viminfo`, and similar files created by MySQL or operating

system utilities. You can safely ignore these error messages or remove the files or directories that cause them if you do not need them.

For nonlocal user management (LDAP, NIS, and so forth), the administrative tools may require additional authentication (such as a password), and will fail if the installing user does not provide this authentication. Even if they fail, the RPM installation will not abort but succeed, and this is intentional. If they failed, some of the intended transfer of ownership may be missing, and it is recommended that the system administrator then manually ensures some appropriate user and group exists and manually transfers ownership following the actions in the RPM spec file.

In MySQL 5.5.31, the RPM spec file has been updated, which has the following consequences:

- For a non-upgrade installation (no existing MySQL version installed), it possible to install MySQL using [yum](#).
- For upgrades, it is necessary to clean up any earlier MySQL installations. In effect, the update is performed by removing the old installations and installing the new one.

Additional details follow.

For a non-upgrade installation of MySQL 5.5.31, it is possible to install using [yum](#):

```
shell> yum install MySQL-server-NEWVERSION.glibc23.i386.rpm
```

For upgrades to MySQL 5.5.31, the upgrade is performed by removing the old installation and installing the new one. To do this, use the following procedure:

1. Remove the existing 5.5.*X* installation. *OLDVERSION* is the version to remove.

```
shell> rpm -e MySQL-server-OLDVERSION.glibc23.i386.rpm
```

Repeat this step for all installed MySQL RPMs.

2. Install the new version. *NEWVERSION* is the version to install.

```
shell> rpm -ivh MySQL-server-NEWVERSION.glibc23.i386.rpm
```

Alternatively, the removal and installation can be done using [yum](#):

```
shell> yum remove MySQL-server-OLDVERSION.glibc23.i386.rpm
shell> yum install MySQL-server-NEWVERSION.glibc23.i386.rpm
```

For some Linux distributions, it might be necessary to increase the limit on number of file descriptors available to `mysqld`. See [File Not Found and Similar Errors](#)

7.2 Installing MySQL on Linux Using Debian Packages

Oracle provides Debian packages for installation on Debian or Debian-like Linux systems. To obtain a package, see [Section 2.2, “How to Get MySQL”](#).

Note

Debian distributions of MySQL are also provided by other vendors. Be aware that they may differ from those built by us in features, capabilities, and conventions (including communication setup), and that the instructions in this manual do not

necessarily apply to installing them. The vendor's instructions should be consulted instead.

Debian package files have names in `mysql-MVER-DVER-CPU.deb` format. *MVER* is the MySQL version and *DVER* is the Debian version. The *CPU* value indicates the processor type or family for which the package is built, as shown in the following table.

Table 7.4 MySQL Installation Packages for Linux CPU Identifiers

CPU Value	Intended Processor Type or Family
i686	Pentium processor or better, 32 bit
x86_64	64-bit x86 processor

After downloading a Debian package, use the following command to install it;

```
shell> dpkg -i mysql-MVER-DVER-CPU.deb
```

The Debian package installs files in the `/opt/mysql/server-5.5` directory.

You may also need to install the `libaio` library if it is not already present on your system:

```
shell> sudo apt-get install libaio1
```

7.3 Installing MySQL on Linux Using Native Package Managers

Many Linux distributions include a version of the MySQL server, client tools, and development components in their native software repositories and can be installed with the platforms' standard package management systems. This section provides basic instructions for installing MySQL using those package management systems.

Important

Native package installations can take care of the download and dependencies required to run MySQL, but the MySQL version will often be some way behind the currently available release. You will also normally be unable to install development releases, as these are not usually made available in the native repository.

Distribution specific instructions are shown below:

- **Red Hat Linux, Fedora, CentOS**

For Red Hat and similar distributions, the MySQL distribution is divided into a number of separate packages, `mysql` for the client tools, `mysql-server` for the server and associated tools, and `mysql-libs` for the libraries. The libraries are required if you want to provide connectivity from different languages and environments such as Perl, Python and others.

To install, use the `yum` command to specify the packages that you want to install. For example:

```
root-shell> yum install mysql mysql-server mysql-libs mysql-server
Loaded plugins: presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package mysql.x86_64 0:5.1.48-2.fc13 set to be updated
--> Package mysql-libs.x86_64 0:5.1.48-2.fc13 set to be updated
--> Package mysql-server.x86_64 0:5.1.48-2.fc13 set to be updated
```

```

--> Processing Dependency: perl-DBD-MySQL for package: mysql-server-5.1.48-2.fc13.x86_64
--> Running transaction check
----> Package perl-DBD-MySQL.x86_64 0:4.017-1.fc13 set to be updated
--> Finished Dependency Resolution
Dependencies Resolved
=====
Package                Arch          Version           Repository        Size
=====
Installing:
mysql                  x86_64        5.1.48-2.fc13    updates          889 k
mysql-libs             x86_64        5.1.48-2.fc13    updates          1.2 M
mysql-server          x86_64        5.1.48-2.fc13    updates          8.1 M
Installing for dependencies:
perl-DBD-MySQL        x86_64        4.017-1.fc13     updates          136 k
Transaction Summary
=====
Install      4 Package(s)
Upgrade     0 Package(s)
Total download size: 10 M
Installed size: 30 M
Is this ok [y/N]: y
Downloading Packages:
Setting up and reading Presto delta metadata
Processing delta metadata
Package(s) data still to download: 10 M
(1/4): mysql-5.1.48-2.fc13.x86_64.rpm | 889 kB   00:04
(2/4): mysql-libs-5.1.48-2.fc13.x86_64.rpm | 1.2 MB  00:06
(3/4): mysql-server-5.1.48-2.fc13.x86_64.rpm | 8.1 MB  00:40
(4/4): perl-DBD-MySQL-4.017-1.fc13.x86_64.rpm | 136 kB  00:00
-----
Total                               201 kB/s | 10 MB   00:52
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing      : mysql-libs-5.1.48-2.fc13.x86_64          1/4
  Installing      : mysql-5.1.48-2.fc13.x86_64             2/4
  Installing      : perl-DBD-MySQL-4.017-1.fc13.x86_64     3/4
  Installing      : mysql-server-5.1.48-2.fc13.x86_64      4/4
Installed:
mysql.x86_64 0:5.1.48-2.fc13          mysql-libs.x86_64 0:5.1.48-2.fc13
mysql-server.x86_64 0:5.1.48-2.fc13
Dependency Installed:
perl-DBD-MySQL.x86_64 0:4.017-1.fc13
Complete!

```

MySQL and the MySQL server should now be installed. A sample configuration file is installed into `/etc/my.cnf`. An init script, to start and stop the server, will have been installed into `/etc/init.d/mysqld`. To start the MySQL server use `service`:

```
root-shell> service mysqld start
```

To enable the server to be started and stopped automatically during boot, use `chkconfig`:

```
root-shell> chkconfig --levels 235 mysqld on
```

Which enables the MySQL server to be started (and stopped) automatically at the specified the run levels.

The database tables will have been automatically created for you, if they do not already exist. You should, however, run `mysql_secure_installation` to set the root passwords on your server.

- **Debian, Ubuntu, Kubuntu**

On Debian and related distributions, there are two packages, `mysql-client` and `mysql-server`, for the client and server components respectively. You should specify an explicit version, for example `mysql-client-5.1`, to ensure that you install the version of MySQL that you want.

To download and install, including any dependencies, use the `apt-get` command, specifying the packages that you want to install.

Note

Before installing, make sure that you update your `apt-get` index files to ensure you are downloading the latest available version.

A sample installation of the MySQL packages might look like this (some sections trimmed for clarity):

```
root-shell> apt-get install mysql-client-5.1 mysql-server-5.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.28-11 linux-headers-2.6.28-11-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-common postfix
Suggested packages:
  dbshell libipc-sharedcache-perl tinyca procmail postfix-mysql postfix-pgsql
  postfix-ldap postfix-pcre sasl2-bin resolvconf postfix-cdb
The following NEW packages will be installed
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-client-5.1 mysql-common mysql-server-5.1 postfix
0 upgraded, 13 newly installed, 0 to remove and 182 not upgraded.
Need to get 1907kB/25.3MB of archives.
After this operation, 59.5MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get: 1 http://gb.archive.ubuntu.com jaunty-updates/main mysql-common 5.1.30really5.0.75-0ubuntu10.5 [63.
Get: 2 http://gb.archive.ubuntu.com jaunty-updates/main libmysqlclient15off 5.1.30really5.0.75-0ubuntu10
Fetched 1907kB in 9s (205kB/s)
Preconfiguring packages ...
Selecting previously deselected package mysql-common.
(Reading database ... 121260 files and directories currently installed.)
...
Processing 1 added doc-base file(s)...
Registering documents with scrollkeeper...
Setting up libnet-daemon-perl (0.43-1) ...
Setting up libplrpc-perl (0.2020-1) ...
Setting up libdbi-perl (1.607-1) ...
Setting up libmysqlclient15off (5.1.30really5.0.75-0ubuntu10.5) ...
Setting up libdbd-mysql-perl (4.008-1) ...
Setting up libmysqlclient16 (5.1.31-lubuntu2) ...
Setting up mysql-client-5.1 (5.1.31-lubuntu2) ...
Setting up mysql-server-5.1 (5.1.31-lubuntu2) ...
  * Stopping MySQL database server mysqld
  ...done.
100825 11:46:15 InnoDB: Started; log sequence number 0 46409
100825 11:46:15 InnoDB: Starting shutdown...
100825 11:46:17 InnoDB: Shutdown completed; log sequence number 0 46409
100825 11:46:17 [Warning] Forcing shutdown of 1 plugins
  * Starting MySQL database server mysqld
  ...done.
  * Checking for corrupt, not cleanly closed and upgrade needing tables.
...
```

```
Processing triggers for libc6 ...  
ldconfig deferred processing now taking place
```

Note

The `apt-get` command will install a number of packages, including the MySQL server, in order to provide the typical tools and application environment. This can mean that you install a large number of packages in addition to the main MySQL package.

During installation, the initial database will be created, and you will be prompted for the MySQL root password (and confirmation). A configuration file will have been created in `/etc/mysql/my.cnf`. An init script will have been created in `/etc/init.d/mysql`.

The server will already be started. You can manually start and stop the server using:

```
root-shell> service mysql [start|stop]
```

The service will automatically be added to the 2, 3 and 4 run levels, with stop scripts in the single, shutdown and restart levels.

• Gentoo Linux

As a source-based distribution, installing MySQL on Gentoo involves downloading the source, patching the Gentoo specifics, and then compiling the MySQL server and installing it. This process is handled automatically by the `emerge` command.

The MySQL server and client tools are provided within a single package, `dev-db/mysql`. You can obtain a list of the versions available to install by looking at the portage directory for the package:

```
root-shell> ls /usr/portage/dev-db/mysql/mysql-5.5*  
mysql-5.5.46.ebuild  
mysql-5.5.47.ebuild
```

To install a specific MySQL version, you must specify the entire atom. For example:

```
root-shell> emerge =dev-db/mysql-5.5.46
```

After installation, you should initialize the data directory and set the password for the MySQL `root` user (see [Section 9.1, "Initializing the Data Directory"](#)). Alternatively, use the configuration interface to perform those tasks:

```
root-shell> emerge --config =dev-db/mysql-5.5.46
```

During installation, a sample configuration file is created for you in `/etc/mysql/my.cnf`, and an init script is created in `/etc/init.d/mysql`.

To enable MySQL to start automatically at the normal (default) run levels, use this command:

```
root-shell> rc-update add mysql default
```

Chapter 8 Installing MySQL on Solaris

Table of Contents

8.1 Installing MySQL on Solaris Using a Solaris PKG 150

Note

MySQL 5.5 supports Solaris 10 (Update 11 and later), and Solaris 11 (Update 3 and later).

MySQL on Solaris is available in a number of different formats.

- For information on installing using the native Solaris PKG format, see [Section 8.1, “Installing MySQL on Solaris Using a Solaris PKG”](#).
- To use a standard `tar` binary installation, use the notes provided in [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#). Check the notes and hints at the end of this section for Solaris specific notes that you may need before or after installation.

To obtain a binary MySQL distribution for Solaris in tarball or PKG format, <http://dev.mysql.com/downloads/mysql/5.5.html>.

Additional notes to be aware of when installing and using MySQL on Solaris:

- If you want to use MySQL with the `mysql` user and group, use the `groupadd` and `useradd` commands:

```
groupadd mysql
useradd -g mysql -s /bin/false mysql
```

- If you install MySQL using a binary tarball distribution on Solaris, because the Solaris `tar` cannot handle long file names, use GNU `tar` (`gtar`) to unpack the distribution. If you do not have GNU `tar` on your system, install it with the following command:

```
pkg install archiver/gnu-tar
```

- You should mount any file systems on which you intend to store `InnoDB` files with the `forcedirectio` option. (By default mounting is done without this option.) Failing to do so will cause a significant drop in performance when using the `InnoDB` storage engine on this platform.
- If you would like MySQL to start automatically, you can copy `support-files/mysql.server` to `/etc/init.d` and create a symbolic link to it named `/etc/rc3.d/S99mysql.server`.
- If too many processes try to connect very rapidly to `mysqld`, you should see this error in the MySQL log:

```
Error in accept: Protocol error
```

You might try starting the server with the `--back_log=50` option as a workaround for this.

- To configure the generation of core files on Solaris you should use the `coreadm` command. Because of the security implications of generating a core on a `setuid()` application, by default, Solaris does not support core files on `setuid()` programs. However, you can modify this behavior using `coreadm`.

If you enable `setuid()` core files for the current user, they will be generated using the mode 600 and owned by the superuser.

8.1 Installing MySQL on Solaris Using a Solaris PKG

You can install MySQL on Solaris using a binary package using the native Solaris PKG format instead of the binary tarball distribution.

To use this package, download the corresponding `mysql-VERSION-solaris10-PLATFORM.pkg.gz` file, then uncompress it. For example:

```
shell> gunzip mysql-5.5.58-solaris10-x86_64.pkg.gz
```

To install a new package, use `pkgadd` and follow the onscreen prompts. You must have root privileges to perform this operation:

```
shell> pkgadd -d mysql-5.5.58-solaris10-x86_64.pkg
The following packages are available:
  1  mysql      MySQL Community Server (GPL)
      (i86pc) 5.5.58
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:
```

The PKG installer installs all of the files and tools needed, and then initializes your database if one does not exist. To complete the installation, you should set the root password for MySQL as provided in the instructions at the end of the installation. Alternatively, you can run the `mysql_secure_installation` script that comes with the installation.

By default, the PKG package installs MySQL under the root path `/opt/mysql`. You can change only the installation root path when using `pkgadd`, which can be used to install MySQL in a different Solaris zone. If you need to install in a specific directory, use a binary `tar` file distribution.

The `pkg` installer copies a suitable startup script for MySQL into `/etc/init.d/mysql`. To enable MySQL to startup and shutdown automatically, you should create a link between this file and the init script directories. For example, to ensure safe startup and shutdown of MySQL you could use the following commands to add the right links:

```
shell> ln /etc/init.d/mysql /etc/rc3.d/S91mysql
shell> ln /etc/init.d/mysql /etc/rc0.d/K02mysql
```

To remove MySQL, the installed package name is `mysql`. You can use this in combination with the `pkgrm` command to remove the installation.

To upgrade when using the Solaris package file format, you must remove the existing installation before installing the updated package. Removal of the package does not delete the existing database information, only the server, binaries and support files. The typical upgrade sequence is therefore:

```
shell> mysqladmin shutdown
shell> pkgrm mysql
shell> pkgadd -d mysql-5.5.58-solaris10-x86_64.pkg
shell> mysql_safe &
shell> mysql_upgrade
```

You should check the notes in [Chapter 10, Upgrading or Downgrading MySQL](#) before performing any upgrade.

Chapter 9 Postinstallation Setup and Testing

Table of Contents

9.1 Initializing the Data Directory	151
9.1.1 Problems Running <code>mysql_install_db</code>	153
9.2 Starting the Server	155
9.2.1 Troubleshooting Problems Starting the MySQL Server	156
9.3 Testing the Server	158
9.4 Securing the Initial MySQL Accounts	160
9.5 Starting and Stopping MySQL Automatically	164

This section discusses tasks that you should perform after installing MySQL:

- If necessary, initialize the data directory and create the MySQL grant tables. For some MySQL installation methods, data directory initialization may be done for you automatically:
 - Installation on Windows
 - Installation on Linux using a server RPM distribution.
 - Installation using the native packaging system on many platforms, including Debian Linux, Ubuntu Linux, Gentoo Linux, and others.
 - Installation on OS X using a DMG distribution.

For other platforms and installation types, including installation from generic binary and source distributions, you must initialize the data directory yourself. For instructions, see [Section 9.1, “Initializing the Data Directory”](#).

- For instructions, see [Section 9.2, “Starting the Server”](#), and [Section 9.3, “Testing the Server”](#).
- Assign passwords to any initial accounts in the grant tables, if that was not already done during data directory initialization. Passwords prevent unauthorized access to the MySQL server. You may also wish to restrict access to test databases. For instructions, see [Section 9.4, “Securing the Initial MySQL Accounts”](#).
- Optionally, arrange for the server to start and stop automatically when your system starts and stops. For instructions, see [Section 9.5, “Starting and Stopping MySQL Automatically”](#).
- Optionally, populate time zone tables to enable recognition of named time zones. For instructions, see [MySQL Server Time Zone Support](#).

When you are ready to create additional user accounts, you can find information on the MySQL access control system and account management in [The MySQL Access Privilege System](#), and [MySQL User Account Management](#).

9.1 Initializing the Data Directory

After installing MySQL, you must initialize the data directory, including the tables in the `mysql` system database. For some MySQL installation methods, data directory initialization may be done automatically, as described in [Chapter 9, *Postinstallation Setup and Testing*](#). For other installation methods, including installation from generic binary and source distributions, you must initialize the data directory yourself.

This section describes how to initialize the data directory on Unix and Unix-like systems. (For Windows, see [Section 5.9, “Windows Postinstallation Procedures”](#).) For some suggested commands that you can use to test whether the server is accessible and working properly, see [Section 9.3, “Testing the Server”](#).

In the examples shown here, the server runs under the user ID of the `mysql` login account. This assumes that such an account exists. Either create the account if it does not exist, or substitute the name of a different existing login account that you plan to use for running the server. For information about creating the account, see [Creating a `mysql` System User and Group](#), in [Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*](#).

1. Change location into the top-level directory of your MySQL installation, represented here by `BASEDIR`:

```
shell> cd BASEDIR
```

`BASEDIR` is likely to be something like `/usr/local/mysql` or `/usr/local`. The following steps assume that you have changed location to this directory.

You will find several files and subdirectories in the `BASEDIR` directory. The most important for installation purposes are the `bin` and `scripts` subdirectories, which contain the server as well as client and utility programs.

2. If necessary, ensure that the distribution contents are accessible to `mysql`. If you installed the distribution as `mysql`, no further action is required. If you installed the distribution as `root`, its contents will be owned by `root`. Change its ownership to `mysql` by executing the following commands as `root` in the installation directory. The first command changes the owner attribute of the files to the `mysql` user. The second changes the group attribute to the `mysql` group.

```
shell> chown -R mysql .
shell> chgrp -R mysql .
```

3. If necessary, initialize the data directory, including the `mysql` database containing the initial MySQL grant tables that determine how users are permitted to connect to the server.

Typically, data directory initialization need be done only the first time you install MySQL. If you are upgrading an existing installation, you should run `mysql_upgrade` instead (see [mysql_upgrade — Check and Upgrade MySQL Tables](#)). However, the command that initializes the data directory does not overwrite any existing privilege tables, so it should be safe to run in any circumstances.

```
shell> scripts/mysql_install_db --user=mysql
```

It is important to make sure that the database directories and files are owned by the `mysql` login account so that the server has read and write access to them when you run it later. To ensure this if you run `mysql_install_db` as `root`, include the `--user` option as shown. Otherwise, you should execute the program while logged in as `mysql`, in which case you can omit the `--user` option from the command.

The `mysql_install_db` command creates the server's data directory. Under the data directory, it creates directories for the `mysql` database that holds the grant tables and the `test` database that you can use to test MySQL. The program also creates privilege table entries for the initial account or accounts. `test_.` For a complete listing and description of the grant tables, see [The MySQL Access Privilege System](#).

It might be necessary to specify other options such as `--basedir` or `--datadir` if `mysql_install_db` does not identify the correct locations for the installation directory or data directory. For example:

```
shell> scripts/mysql_install_db --user=mysql \  
      --basedir=/opt/mysql/mysql \  
      --datadir=/opt/mysql/mysql/data
```

If you do not want to have the `test` database, you can remove it after starting the server, using the instructions in [Section 9.4, “Securing the Initial MySQL Accounts”](#).

If you have trouble with `mysql_install_db` at this point, see [Section 9.1.1, “Problems Running `mysql_install_db`”](#).

4. After initializing the data directory, you can establish the final installation ownership settings. To leave the installation owned by `mysql`, no action is required here. Otherwise, most of the MySQL installation can be owned by `root` if you like. The exception is that the data directory must be owned by `mysql`. To accomplish this, run the following commands as `root` in the installation directory. For some distribution types, the data directory might be named `var` rather than `data`; adjust the second command accordingly.

```
shell> chown -R root .  
shell> chown -R mysql data
```

If the plugin directory (the directory named by the `plugin_dir` system variable) is writable by the server, it may be possible for a user to write executable code to a file in the directory using `SELECT ... INTO DUMPFILE`. This can be prevented by making the plugin directory read only to the server or by setting the `secure_file_priv` system variable at server startup to a directory where `SELECT` writes can be performed safely.

5. If you installed MySQL using a source distribution, you may want to optionally copy one of the provided configuration files from the `support-files` directory into your `/etc` directory. There are different sample configuration files for different use cases, server types, and CPU and RAM configurations. To use one of these standard files, copy it to `/etc/my.cnf`, or `/etc/mysql/my.cnf` and edit and check the configuration before starting your MySQL server for the first time.

You can also create `my.cnf` yourself and place into it the options the server should use at startup. See [Server Configuration Defaults](#).

If you do not copy one of the standard configuration files or create your own, the MySQL server starts with its default settings.

6. If you want MySQL to start automatically when you boot your machine, see [Section 9.5, “Starting and Stopping MySQL Automatically”](#).

Data directory initialization creates time zone tables in the `mysql` database but does not populate them. To do so, use the instructions in [MySQL Server Time Zone Support](#).

9.1.1 Problems Running `mysql_install_db`

The purpose of the `mysql_install_db` program is to initialize the data directory, including the tables in the `mysql` system database. It does not overwrite existing MySQL privilege tables, and it does not affect any other data.

To re-create your privilege tables, first stop the `mysqld` server if it is running. Then rename the `mysql` directory under the data directory to save it, and run `mysql_install_db`. Suppose that your current directory is the MySQL installation directory and that `mysql_install_db` is located in the `bin` directory and the data directory is named `data`. To rename the `mysql` database and re-run `mysql_install_db`, use these commands.

```
shell> mv data/mysql data/mysql.old
shell> scripts/mysql_install_db --user=mysql
```

When you run `mysql_install_db`, you might encounter the following problems:

- **`mysql_install_db` fails to install the grant tables**

You may find that `mysql_install_db` fails to install the grant tables and terminates after displaying the following messages:

```
Starting mysqld daemon with databases from XXXXXX
mysqld ended
```

In this case, you should examine the error log file very carefully. The log should be located in the directory `XXXXXX` named by the error message and should indicate why `mysqld` did not start. If you do not understand what happened, include the log when you post a bug report. See [How to Report Bugs or Problems](#).

- **There is a `mysqld` process running**

This indicates that the server is running, in which case the grant tables have probably been created already. If so, there is no need to run `mysql_install_db` at all because it needs to be run only once, when you first install MySQL.

- **Installing a second `mysqld` server does not work when one server is running**

This can happen when you have an existing MySQL installation, but want to put a new installation in a different location. For example, you might have a production installation, but you want to create a second installation for testing purposes. Generally the problem that occurs when you try to run a second server is that it tries to use a network interface that is in use by the first server. In this case, you should see one of the following error messages:

```
Can't start server: Bind on TCP/IP port:
Address already in use
Can't start server: Bind on unix socket...
```

For instructions on setting up multiple servers, see [Running Multiple MySQL Instances on One Machine](#).

- **You do not have write access to the `/tmp` directory**

If you do not have write access to create temporary files or a Unix socket file in the default location (the `/tmp` directory) or the `TMPDIR` environment variable, if it has been set, an error occurs when you run `mysql_install_db` or the `mysqld` server.

You can specify different locations for the temporary directory and Unix socket file by executing these commands prior to starting `mysql_install_db` or `mysqld`, where `some_tmp_dir` is the full path name to some directory for which you have write permission:

```
shell> TMPDIR=/some_tmp_dir/
shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysql.sock
shell> export TMPDIR MYSQL_UNIX_PORT
```

Then you should be able to run `mysql_install_db` and start the server with these commands:

```
shell> scripts/mysql_install_db --user=mysql
```



```
shell> bin/mysqld_safe --user=mysql &
```

If `mysql_install_db` is located in the `scripts` directory, modify the first command to `scripts/mysql_install_db`.

See [How to Protect or Change the MySQL Unix Socket File](#), and [Chapter 11, MySQL Program Environment Variables](#).

There are some alternatives to running the `mysql_install_db` program provided in the MySQL distribution:

- If you want the initial privileges to be different from the standard defaults, use account-management statements such as `CREATE USER`, `GRANT`, and `REVOKE` to change the privileges *after* the grant tables have been set up. In other words, run `mysql_install_db`, and then use `mysql -u root mysql` to connect to the server as the MySQL `root` user so that you can issue the necessary statements. (See [Account Management Statements](#).)

To install MySQL on several machines with the same privileges, put the `CREATE USER`, `GRANT`, and `REVOKE` statements in a file and execute the file as a script using `mysql` after running `mysql_install_db`. For example:

```
shell> scripts/mysql_install_db --user=mysql
shell> bin/mysql -u root < your_script_file
```

This enables you to avoid issuing the statements manually on each machine.

- It is possible to re-create the grant tables completely after they have previously been created. You might want to do this if you are just learning how to use `CREATE USER`, `GRANT`, and `REVOKE` and have made so many modifications after running `mysql_install_db` that you want to wipe out the tables and start over.

To re-create the grant tables, stop the server if it is running and remove the `mysql` database directory. Then run `mysql_install_db` again.

9.2 Starting the Server

This section describes how start the server on Unix and Unix-like systems. (For Windows, see [Section 5.7.4, “Starting the Server for the First Time”](#).) For some suggested commands that you can use to test whether the server is accessible and working properly, see [Section 9.3, “Testing the Server”](#).

Start the MySQL server like this:

```
shell> bin/mysqld_safe --user=mysql &
```

It is important that the MySQL server be run using an unprivileged (non-`root`) login account. To ensure this if you run `mysqld_safe` as `root`, include the `--user` option as shown. Otherwise, execute the program while logged in as `mysql`, in which case you can omit the `--user` option from the command.

For further instructions for running MySQL as an unprivileged user, see [How to Run MySQL as a Normal User](#).

If the command fails immediately and prints `mysqld ended`, look for information in the error log (which by default is the `host_name.err` file in the data directory).

If the server is unable to access the data directory it starts or read the grant tables in the `mysql` database, it writes a message to its error log. Such problems can occur if you neglected to create the grant tables by

initializing the data directory before proceeding to this step, or if you ran the command that initializes the data directory without the `--user` option. Remove the `data` directory and run the command with the `--user` option.

If you have other problems starting the server, see [Section 9.2.1, “Troubleshooting Problems Starting the MySQL Server”](#). For more information about `mysqld_safe`, see [mysqld_safe — MySQL Server Startup Script](#).

You can set up new accounts using the `bin/mysql_setpermission` script if you install the `DBI` and `DBD: :mysql` Perl modules. See [mysql_setpermission — Interactively Set Permissions in Grant Tables](#). For Perl module installation instructions, see [Chapter 12, Perl Installation Notes](#).

If you would like to use `mysqlaccess` and have the MySQL distribution in some nonstandard location, you must change the location where `mysqlaccess` expects to find the `mysql` client. Edit the `bin/mysqlaccess` script at approximately line 18. Search for a line that looks like this:

```
$MYSQL = '/usr/local/bin/mysql'; # path to mysql executable
```

Change the path to reflect the location where `mysql` actually is stored on your system. If you do not do this, a [Broken pipe](#) error will occur when you run `mysqlaccess`.

9.2.1 Troubleshooting Problems Starting the MySQL Server

This section provides troubleshooting suggestions for problems starting the server. For additional suggestions for Windows systems, see [Section 5.8, “Troubleshooting a Microsoft Windows MySQL Server Installation”](#).

If you have problems starting the server, here are some things to try:

- Check the error log to see why the server does not start.
- Specify any special options needed by the storage engines you are using.
- Make sure that the server knows where to find the data directory.
- Make sure that the server can access the data directory. The ownership and permissions of the data directory and its contents must be set such that the server can read and modify them.
- Verify that the network interfaces the server wants to use are available.

Some storage engines have options that control their behavior. You can create a `my.cnf` file and specify startup options for the engines that you plan to use. If you are going to use storage engines that support transactional tables ([InnoDB](#), [NDB](#)), be sure that you have them configured the way you want before starting the server:

If you are using [InnoDB](#) tables, see [InnoDB Configuration](#).

Storage engines will use default option values if you specify none, but it is recommended that you review the available options and specify explicit values for those for which the defaults are not appropriate for your installation.

When the `mysqld` server starts, it changes location to the data directory. This is where it expects to find databases and where it expects to write log files. The server also writes the pid (process ID) file in the data directory.

The data directory location is hardwired in when the server is compiled. This is where the server looks for the data directory by default. If the data directory is located somewhere else on your system, the server will

not work properly. You can determine what the default path settings are by invoking `mysqld` with the `--verbose` and `--help` options.

If the default locations do not match the MySQL installation layout on your system, you can override them by specifying options to `mysqld` or `mysqld_safe` on the command line or in an option file.

To specify the location of the data directory explicitly, use the `--datadir` option. However, normally you can tell `mysqld` the location of the base directory under which MySQL is installed and it looks for the data directory there. You can do this with the `--basedir` option.

To check the effect of specifying path options, invoke `mysqld` with those options followed by the `--verbose` and `--help` options. For example, if you change location into the directory where `mysqld` is installed and then run the following command, it shows the effect of starting the server with a base directory of `/usr/local`:

```
shell> ./mysqld --basedir=/usr/local --verbose --help
```

You can specify other options such as `--datadir` as well, but `--verbose` and `--help` must be the last options.

Once you determine the path settings you want, start the server without `--verbose` and `--help`.

If `mysqld` is currently running, you can find out what path settings it is using by executing this command:

```
shell> mysqladmin variables
```

Or:

```
shell> mysqladmin -h host_name variables
```

`host_name` is the name of the MySQL server host.

If you get `Errcode 13` (which means `Permission denied`) when starting `mysqld`, this means that the privileges of the data directory or its contents do not permit server access. In this case, you change the permissions for the involved files and directories so that the server has the right to use them. You can also start the server as `root`, but this raises security issues and should be avoided.

Change location into the data directory and check the ownership of the data directory and its contents to make sure the server has access. For example, if the data directory is `/usr/local/mysql/var`, use this command:

```
shell> ls -la /usr/local/mysql/var
```

If the data directory or its files or subdirectories are not owned by the login account that you use for running the server, change their ownership to that account. If the account is named `mysql`, use these commands:

```
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql/var
```

Even with correct ownership, MySQL might fail to start up if there is other security software running on your system that manages application access to various parts of the file system. In this case, reconfigure that software to enable `mysqld` to access the directories it uses during normal operation.

If the server fails to start up correctly, check the error log. Log files are located in the data directory (typically `C:\Program Files\MySQL\MySQL Server 5.5\data` on Windows, `/usr/local/mysql/`

`data` for a Unix/Linux binary distribution, and `/usr/local/var` for a Unix/Linux source distribution). Look in the data directory for files with names of the form `host_name.err` and `host_name.log`, where `host_name` is the name of your server host. Then examine the last few lines of these files. You can use `tail` to display them:

```
shell> tail host_name.err
shell> tail host_name.log
```

The error log should contain information that indicates why the server could not start.

If either of the following errors occur, it means that some other program (perhaps another `mysqld` server) is using the TCP/IP port or Unix socket file that `mysqld` is trying to use:

```
Can't start server: Bind on TCP/IP port: Address already in use
Can't start server: Bind on unix socket...
```

Use `ps` to determine whether you have another `mysqld` server running. If so, shut down the server before starting `mysqld` again. (If another server is running, and you really want to run multiple servers, you can find information about how to do so in [Running Multiple MySQL Instances on One Machine](#).)

If no other server is running, try to execute the command `telnet your_host_name tcp_ip_port_number`. (The default MySQL port number is 3306.) Then press Enter a couple of times. If you do not get an error message like `telnet: Unable to connect to remote host: Connection refused`, some other program is using the TCP/IP port that `mysqld` is trying to use. You will need to track down what program this is and disable it, or else tell `mysqld` to listen to a different port with the `--port` option. In this case, you will also need to specify the port number for client programs when connecting to the server using TCP/IP.

Another reason the port might be inaccessible is that you have a firewall running that blocks connections to it. If so, modify the firewall settings to permit access to the port.

If the server starts but you cannot connect to it, you should make sure that you have an entry in `/etc/hosts` that looks like this:

```
127.0.0.1      localhost
```

If you cannot get `mysqld` to start, you can try to make a trace file to find the problem by using the `--debug` option. See [The DEBUG Package](#).

9.3 Testing the Server

After the data directory is initialized and you have started the server, perform some simple tests to make sure that it works satisfactorily. This section assumes that your current location is the MySQL installation directory and that it has a `bin` subdirectory containing the MySQL programs used here. If that is not true, adjust the command path names accordingly.

Alternatively, add the `bin` directory to your `PATH` environment variable setting. That enables your shell (command interpreter) to find MySQL programs properly, so that you can run a program by typing only its name, not its path name. See [Setting Environment Variables](#).

Use `mysqladmin` to verify that the server is running. The following commands provide simple tests to check whether the server is up and responding to connections:

```
shell> bin/mysqladmin version
```

```
shell> bin/mysqladmin variables
```

If you cannot connect to the server, specify a `-u root` option to connect as `root`. If you have assigned a password for the `root` account already, you'll also need to specify `-p` on the command line and enter the password when prompted. For example:

```
shell> bin/mysqladmin -u root -p version
Enter password: (enter root password here)
```

The output from `mysqladmin version` varies slightly depending on your platform and version of MySQL, but should be similar to that shown here:

```
shell> bin/mysqladmin version
mysqladmin Ver 14.12 Distrib 5.5.58, for pc-linux-gnu on i686
...
Server version          5.5.58
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/lib/mysql/mysql.sock
Uptime:                 14 days 5 hours 5 min 21 sec
Threads: 1  Questions: 366  Slow queries: 0
Opens: 0  Flush tables: 1  Open tables: 19
Queries per second avg: 0.000
```

To see what else you can do with `mysqladmin`, invoke it with the `--help` option.

Verify that you can shut down the server (include a `-p` option if the `root` account has a password already):

```
shell> bin/mysqladmin -u root shutdown
```

Verify that you can start the server again. Do this by using `mysqld_safe` or by invoking `mysqld` directly. For example:

```
shell> bin/mysqld_safe --user=mysql &
```

If `mysqld_safe` fails, see [Section 9.2.1, "Troubleshooting Problems Starting the MySQL Server"](#).

Run some simple tests to verify that you can retrieve information from the server. The output should be similar to that shown here.

Use `mysqlshow` to see what databases exist:

```
shell> bin/mysqlshow
+-----+
| Databases |
+-----+
| information_schema |
| mysql           |
| performance_schema |
| test           |
+-----+
```

The list of installed databases may vary, but will always include the minimum of `mysql` and `information_schema`.

If you specify a database name, `mysqlshow` displays a list of the tables within the database:

```
shell> bin/mysqlshow mysql
```

```

Database: mysql
+-----+
|           Tables           |
+-----+
| columns_priv              |
| db                        |
| event                    |
| func                     |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| ndb_binlog_index          |
| plugin                    |
| proc                     |
| procs_priv                |
| proxies_priv              |
| servers                   |
| slow_log                  |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+-----+

```

Use the `mysql` program to select information from a table in the `mysql` database:

```

shell> bin/mysql -e "SELECT User, Host, plugin FROM mysql.user" mysql
+-----+-----+-----+
| User | Host      | plugin                |
+-----+-----+-----+
| root | localhost | mysql_native_password |
+-----+-----+-----+

```

At this point, your server is running and you can access it. To tighten security if you have not yet assigned passwords to the initial account or accounts, follow the instructions in [Section 9.4, “Securing the Initial MySQL Accounts”](#).

For more information about `mysql`, `mysqladmin`, and `mysqlshow`, see [mysql — The MySQL Command-Line Tool](#), [mysqladmin — Client for Administering a MySQL Server](#), and [mysqlshow — Display Database, Table, and Column Information](#).

9.4 Securing the Initial MySQL Accounts

The MySQL installation process involves initializing the data directory, including the `mysql` database containing the grant tables that define MySQL accounts. For details, see [Chapter 9, Postinstallation Setup and Testing](#).

This section describes how to assign passwords to the initial accounts created during the MySQL installation procedure, if you have not already done so.

The `mysql.user` grant table defines the initial MySQL user accounts and their access privileges:

- Some accounts have the user name `root`. These are superuser accounts that have all privileges and can do anything. If these `root` accounts have empty passwords, anyone can connect to the MySQL server as `root` *without a password* and be granted all privileges.

- On Windows, `root` accounts are created that permit connections from the local host only. Connections can be made by specifying the host name `localhost`, the IP address `127.0.0.1`, or the IPv6 address `::1`. If the user selects the **Enable root access from remote machines** option during installation, the Windows installer creates another `root` account that permits connections from any host.
- On Unix, each `root` account permits connections from the local host. Connections can be made by specifying the host name `localhost`, the IP address `127.0.0.1`, the IPv6 address `::1`, or the actual host name or IP address.

An attempt to connect to the host `127.0.0.1` normally resolves to the `localhost` account. However, this fails if the server is run with the `--skip-name-resolve` option, so the `127.0.0.1` account is useful in that case. The `::1` account is used for IPv6 connections.

- If accounts for anonymous users were created, these have an empty user name. The anonymous accounts have no password, so anyone can use them to connect to the MySQL server.
- On Windows, there is one anonymous account that permits connections from the local host. Connections can be made by specifying a host name of `localhost`.
- On Unix, each anonymous account permits connections from the local host. Connections can be made by specifying a host name of `localhost` for one of the accounts, or the actual host name or IP address for the other.
- The `'root'@'localhost'` account also has a row in the `mysql.proxies_priv` table that enables granting the `PROXY` privilege for `' '@'`, that is, for all users and all hosts. This enables `root` to set up proxy users, as well as to delegate to other accounts the authority to set up proxy users. See [Proxy Users](#).

To display which accounts exist in the `mysql.user` table and check whether their passwords are empty, use the following statement:

```
mysql> SELECT User, Host, Password FROM mysql.user;
```

User	Host	Password
root	localhost	
root	myhost.example.com	
root	127.0.0.1	
root	::1	
	localhost	
	myhost.example.com	

This output indicates that there are several `root` and anonymous-user accounts, none of which have passwords. The output might differ on your system, but the presence of accounts with empty passwords means that your MySQL installation is unprotected until you do something about it:

- Assign a password to each MySQL `root` account that does not have one.
- To prevent clients from connecting as anonymous users without a password, either assign a password to each anonymous account or remove the accounts.

In addition, the `mysql.db` table contains rows that permit all accounts to access the `test` database and other databases with names that start with `test_`. This is true even for accounts that otherwise have no special privileges such as the default anonymous accounts. This is convenient for testing but inadvisable

on production servers. Administrators who want database access restricted only to accounts that have permissions granted explicitly for that purpose should remove these `mysql.db` table rows.

The following instructions describe how to set up passwords for the initial MySQL accounts, first for the `root` accounts, then for the anonymous accounts. The instructions also cover how to remove anonymous accounts, should you prefer not to permit anonymous access at all, and describe how to remove permissive access to test databases. Replace `new_password` in the examples with the password that you want to use. Replace `host_name` with the name of the server host. You can determine this name from the output of the preceding `SELECT` statement. For the output shown, `host_name` is `myhost.example.com`.

Note

For additional information about setting passwords, see [Assigning Account Passwords](#). If you forget your `root` password after setting it, see [How to Reset the Root Password](#).

To set up additional accounts, see [Adding User Accounts](#).

You might want to defer setting the passwords until later, to avoid the need to specify them while you perform additional setup or testing. However, be sure to set them before using your installation for production purposes.

Note

On Windows, you can also perform the process described in this section using the Configuration Wizard (see [Section 5.6.11, “The Security Options Dialog”](#)). On all platforms, the MySQL distribution includes `mysql_secure_installation`, a command-line utility that automates much of the process of securing a MySQL installation. MySQL Workbench is available on all platforms, and also offers the ability to manage user accounts (see [MySQL Workbench](#)).

Assigning root Account Passwords

A `root` account password can be set several ways. The following discussion demonstrates three methods:

- Use the `SET PASSWORD` statement
- Use the `UPDATE` statement
- Use the `mysqladmin` command-line client program

To assign passwords using `SET PASSWORD`, connect to the server as `root` and issue a `SET PASSWORD` statement for each `root` account listed in the `mysql.user` table.

For Windows, do this:

```
shell> mysql -u root
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'127.0.0.1' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@':::1' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'%' = PASSWORD('new_password');
```

The last statement is unnecessary if the `mysql.user` table has no `root` account with a host value of `%`.

For Unix, do this:


```
shell> mysql -u root
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'127.0.0.1' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@':::1' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'host_name' = PASSWORD('new_password');
```

You can also use a single statement that assigns a password to all `root` accounts by using `UPDATE` to modify the `mysql.user` table directly. This method works on any platform:

```
shell> mysql -u root
mysql> UPDATE mysql.user SET Password = PASSWORD('new_password')
-> WHERE User = 'root';
mysql> FLUSH PRIVILEGES;
```

The `FLUSH` statement causes the server to reread the grant tables. Without it, the password change remains unnoticed by the server until you restart it.

To assign passwords to the `root` accounts using `mysqladmin`, execute the following commands:

```
shell> mysqladmin -u root password "new_password"
shell> mysqladmin -u root -h host_name password "new_password"
```

Those commands apply both to Windows and to Unix. The double quotation marks around the password are not always necessary, but you should use them if the password contains spaces or other characters that are special to your command interpreter.

The `mysqladmin` method of setting the `root` account passwords does not work for the `'root'@'127.0.0.1'` or `'root'@':::1'` account. Use the `SET PASSWORD` method shown earlier.

After the `root` passwords have been set, you must supply the appropriate password whenever you connect as `root` to the server. For example, to shut down the server with `mysqladmin`, use this command:

```
shell> mysqladmin -u root -p shutdown
Enter password: (enter root password here)
```

The `mysql` commands in the following instructions include a `-p` option based on the assumption that you have assigned the `root` account passwords using the preceding instructions and must specify that password when connecting to the server.

Assigning Anonymous Account Passwords

To assign passwords to the anonymous accounts, connect to the server as `root`, then use either `SET PASSWORD` or `UPDATE`.

To use `SET PASSWORD` on Windows, do this:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> SET PASSWORD FOR ''@'localhost' = PASSWORD('new_password');
```

To use `SET PASSWORD` on Unix, do this:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> SET PASSWORD FOR ''@'localhost' = PASSWORD('new_password');
```

```
mysql> SET PASSWORD FOR '@'host_name' = PASSWORD('new_password');
```

To set the anonymous-user account passwords with a single `UPDATE` statement, do this (on any platform):

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> UPDATE mysql.user SET Password = PASSWORD('new_password')
-> WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

The `FLUSH` statement causes the server to reread the grant tables. Without it, the password change remains unnoticed by the server until you restart it.

Removing Anonymous Accounts

If you prefer to remove any anonymous accounts rather than assigning them passwords, do so as follows on Windows:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DROP USER '@'localhost';
```

On Unix, remove the anonymous accounts like this:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DROP USER '@'localhost';
mysql> DROP USER '@'host_name';
```

Securing Test Databases

By default, the `mysql.db` table contains rows that permit access by any user to the `test` database and other databases with names that start with `test_`. (These rows have an empty `User` column value, which for access-checking purposes matches any user name.) This means that such databases can be used even by accounts that otherwise possess no privileges. If you want to remove any-user access to test databases, do so as follows:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DELETE FROM mysql.db WHERE Db LIKE 'test%';
mysql> FLUSH PRIVILEGES;
```

The `FLUSH` statement causes the server to reread the grant tables. Without it, the privilege change remains unnoticed by the server until you restart it.

With the preceding change, only users who have global database privileges or privileges granted explicitly for the `test` database can use it. However, if you prefer that the database not exist at all, drop it:

```
mysql> DROP DATABASE test;
```

9.5 Starting and Stopping MySQL Automatically

This section discusses methods for starting and stopping the MySQL server.

Generally, you start the `mysqld` server in one of these ways:

- Invoke `mysqld` directly. This works on any platform.
- On Windows, you can set up a MySQL service that runs automatically when Windows starts. See [Section 5.7.7, “Starting MySQL as a Windows Service”](#).
- On Unix and Unix-like systems, you can invoke `mysqld_safe`, which tries to determine the proper options for `mysqld` and then runs it with those options. See [mysqld_safe — MySQL Server Startup Script](#).
- On systems that use System V-style run directories (that is, `/etc/init.d` and run-level specific directories), invoke `mysql.server`. This script is used primarily at system startup and shutdown. It usually is installed under the name `mysql`. The `mysql.server` script starts the server by invoking `mysqld_safe`. See [mysql.server — MySQL Server Startup Script](#).
- On OS X, install a launchd daemon to enable automatic MySQL startup at system startup. The daemon starts the server by invoking `mysqld_safe`. For details, see [Section 6.3, “Installing a MySQL Launch Daemon”](#). A MySQL Preference Pane also provides control for starting and stopping MySQL through the System Preferences. See [Section 6.4, “Installing and Using the MySQL Preference Pane”](#).
- On Solaris, use the service management framework (SMF) system to initiate and control MySQL startup.

The `mysqld_safe` and `mysql.server` scripts, Solaris SMF, and the OS X Startup Item (or MySQL Preference Pane) can be used to start the server manually, or automatically at system startup time. `mysql.server` and the Startup Item also can be used to stop the server.

The following table shows which option groups the server and startup scripts read from option files.

Table 9.1 MySQL Startup Scripts and Supported Server Option Groups

Script	Option Groups
<code>mysqld</code>	<code>[mysqld]</code> , <code>[server]</code> , <code>[mysqld-major_version]</code>
<code>mysqld_safe</code>	<code>[mysqld]</code> , <code>[server]</code> , <code>[mysqld_safe]</code>
<code>mysql.server</code>	<code>[mysqld]</code> , <code>[mysql.server]</code> , <code>[server]</code>

`[mysqld-major_version]` means that groups with names like `[mysqld-5.1]` and `[mysqld-5.5]` are read by servers having versions 5.1.x, 5.5.x, and so forth. This feature can be used to specify options that can be read only by servers within a given release series.

For backward compatibility, `mysql.server` also reads the `[mysql_server]` group and `mysqld_safe` also reads the `[safe_mysqld]` group. However, you should update your option files to use the `[mysql.server]` and `[mysqld_safe]` groups instead.

For more information on MySQL configuration files and their structure and contents, see [Using Option Files](#).

Chapter 10 Upgrading or Downgrading MySQL

Table of Contents

10.1 Upgrading MySQL	167
10.1.1 Changes Affecting Upgrades to MySQL 5.5	171
10.2 Downgrading MySQL	177
10.2.1 Changes Affecting Downgrades from MySQL 5.5	181
10.2.2 Downgrading from MySQL Enterprise Edition to MySQL Community Server	181
10.3 Rebuilding or Repairing Tables or Indexes	182
10.4 Copying MySQL Databases to Another Machine	184

This section describes the steps to upgrade or downgrade a MySQL installation.

Upgrading is a common procedure, as you pick up bug fixes within the same MySQL release series or significant features between major MySQL releases. You perform this procedure first on some test systems to make sure everything works smoothly, and then on the production systems.

Downgrading is less common. Typically, you undo an upgrade because of some compatibility or performance issue that occurs on a production system, and was not uncovered during initial upgrade verification on the test systems. As with the upgrade procedure, perform and verify the downgrade procedure on some test systems first, before using it on a production system.

10.1 Upgrading MySQL

This section describes how to upgrade to a new MySQL version.

- [Upgrade Methods](#)
- [Upgrade Paths](#)
- [Before You Begin](#)
- [Performing an In-Place Upgrade](#)
- [Performing a Logical Upgrade](#)
- [Upgrade Troubleshooting](#)

Note

In the following discussion, MySQL commands that must be run using a MySQL account with administrative privileges include `-u root` on the command line to specify the MySQL `root` user. Commands that require a password for `root` also include a `-p` option. Because `-p` is followed by no option value, such commands prompt for the password. Type the password when prompted and press Enter.

SQL statements can be executed using the `mysql` command-line client (connect as `root` to ensure that you have the necessary privileges).

Upgrade Methods

Supported downgrade methods include:

- *In-Place Upgrade*: Involves shutting down the old MySQL version, replacing the old MySQL binaries or packages with the new ones, restarting MySQL on the existing data directory, and running `mysql_upgrade`.
- *Logical Upgrade*: Involves exporting existing data from the old MySQL version using `mysqldump`, installing the new MySQL version, loading the dump file into the new MySQL version, and running `mysql_upgrade`.

Note

MySQL recommends a `mysqldump` upgrade when upgrading from a previous release. For example, use this method when upgrading from 5.1 to 5.5.

For in-place and logical upgrade procedures, see [Performing an In-Place Upgrade](#), and [Performing a Logical Upgrade](#).

If you run MySQL Server on Windows, see [Section 5.10, “Upgrading MySQL on Windows”](#).

Upgrade Paths

- Upgrade is only supported between General Availability (GA) releases.
- Upgrade from MySQL 5.1 to 5.5 is supported. Upgrading to the latest release is recommended before upgrading to the next version. For example, upgrade to the latest MySQL 5.1 release before upgrading to MySQL 5.5.
- Upgrade that skips versions is not supported. For example, upgrading directly from MySQL 5.0 to 5.5 is not supported.
- Upgrade within a release series is supported. For example, upgrading from MySQL 5.5.x to 5.5.y is supported. Skipping a release is also supported. For example, upgrading from MySQL 5.5.x to 5.5.z is supported.

Before You Begin

Before upgrading, review the following information and perform the recommended steps:

- Before upgrading, protect your data by creating a backup of your current databases and log files. The backup should include the `mysql` system database, which contains the MySQL system tables. See [Database Backup Methods](#).
- Review the [Release Notes](#) which provide information about features that are new in the MySQL 5.5 or differ from those found in earlier MySQL releases. Some of these changes may result in incompatibilities.

For a description of MySQL server features that have been removed in MySQL 5.5, see [Features Removed in MySQL 5.5](#). An upgrade requires changes with respect to those features if you use any of them.

- Review [Section 10.1.1, “Changes Affecting Upgrades to MySQL 5.5”](#). This section describes changes that may require action before or after upgrading.
- If you use replication, review [Upgrading a Replication Setup](#).
- If you use XA transactions with InnoDB, run `XA RECOVER` before upgrading to check for uncommitted XA transactions. If results are returned, either commit or rollback the XA transactions by issuing an `XA COMMIT` or `XA ROLLBACK` statement.

- If your MySQL installation contains a large amount of data that might take a long time to convert after an in-place upgrade, you might find it useful to create a “dummy” database instance for assessing what conversions might be needed and the work involved to perform them. Make a copy of your MySQL instance that contains a full copy of the `mysql` database, plus all other databases without data. Run your upgrade procedure on this dummy instance to see what actions might be needed so that you can better evaluate the work involved when performing actual data conversion on your original database instance.
- Rebuilding and reinstalling MySQL language interfaces is recommended whenever you install or upgrade to a new release of MySQL. This applies to MySQL interfaces such as PHP `mysql` extensions, the Perl `DBD: :mysql` module, and the Python `MySQLdb` module.

Performing an In-Place Upgrade

This section describes how to perform an [in-place upgrade](#). Review [Before you Begin](#) before proceeding.

Note

If you upgrade an installation originally produced by installing multiple RPM packages, upgrade all the packages, not just some. For example, if you previously installed the server and client RPMs, do not upgrade just the server RPM.

To perform an in-place upgrade:

1. Review the changes described in [Section 10.1.1, “Changes Affecting Upgrades to MySQL 5.5”](#) for steps to be performed before upgrading.
2. If you use [InnoDB](#), configure MySQL to perform a slow shutdown by setting `innodb_fast_shutdown` to 0. For example:

```
mysql -u root -p --execute="SET GLOBAL innodb_fast_shutdown=0"
```

With a slow shutdown, [InnoDB](#) performs a full purge and change buffer merge before shutting down, which ensures that data files are fully prepared in case of file format differences between releases.

3. Shut down the old MySQL server. For example:

```
mysqladmin -u root -p shutdown
```

4. Upgrade the MySQL binary installation or packages. If upgrading a binary installation, unpack the new MySQL binary distribution package. See [Obtain and Unpack the Distribution](#). For package-based installations, replace the old packages with the new ones.
5. Start the MySQL 5.5 server, using the existing data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir
```

6. Run `mysql_upgrade`. For example:

```
mysql_upgrade -u root -p
```

`mysql_upgrade` examines all tables in all databases for incompatibilities with the current version of MySQL. `mysql_upgrade` also upgrades the `mysql` system database so that you can take advantage of new privileges or capabilities.

Note

`mysql_upgrade` does not upgrade the contents of the help tables. For upgrade instructions, see [Server-Side Help](#).

7. Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir
```

Performing a Logical Upgrade

This section describes how to perform a [logical upgrade](#). Review [Before you Begin](#) before proceeding.

To perform a logical upgrade:

1. Review the changes described in [Section 10.1.1, “Changes Affecting Upgrades to MySQL 5.5”](#) for steps to be performed before upgrading.
2. Export your existing data from the previous MySQL version:

```
mysqldump -u root -p
--add-drop-table --routines --events
--all-databases --force > data-for-upgrade.sql
```

Note

Use the `--routines` and `--events` options with `mysqldump` (as shown above) if your databases include stored programs. The `--all-databases` option includes all databases in the dump, including the `mysql` database that holds the system tables.

3. Shut down the old MySQL server. For example:

```
mysqladmin -u root -p shutdown
```

4. Install MySQL 5.5. For installation instructions, see [Chapter 1, Installing and Upgrading MySQL](#).
5. Initialize a new data directory, as described at [Section 9.1, “Initializing the Data Directory”](#). For example:

```
scripts/mysql_install_db --user=mysql --datadir=/path/to/5.5-datadir
```

6. Start the MySQL 5.5 server, using the new data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/5.5-datadir
```

7. Load the previously created dump file into the new MySQL server. For example:

```
mysql -u root -p --force < data-for-upgrade.sql
```

8. Run `mysql_upgrade`. For example:


```
mysql_upgrade -u root -p
```

`mysql_upgrade` examines all tables in all databases for incompatibilities with the current version of MySQL. `mysql_upgrade` also upgrades the `mysql` system database so that you can take advantage of new privileges or capabilities.

Note

`mysql_upgrade` does not upgrade the contents of the help tables. For upgrade instructions, see [Server-Side Help](#).

9. Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/5.5-datadir
```

Upgrade Troubleshooting

- If problems occur, such as that the new `mysqld` server does not start or that you cannot connect without a password, verify that you do not have an old `my.cnf` file from your previous installation. You can check this with the `--print-defaults` option (for example, `mysqld --print-defaults`). If this command displays anything other than the program name, you have an active `my.cnf` file that affects server or client operation.
- If, after an upgrade, you experience problems with compiled client programs, such as [Commands out of sync](#) or unexpected core dumps, you probably have used old header or library files when compiling your programs. In this case, check the date for your `mysql.h` file and `libmysqlclient.a` library to verify that they are from the new MySQL distribution. If not, recompile your programs with the new headers and libraries. Recompilation might also be necessary for programs compiled against the shared client library if the library major version number has changed (for example, from `libmysqlclient.so.15` to `libmysqlclient.so.16`).
- If you have created a user-defined function (UDF) with a given name and upgrade MySQL to a version that implements a new built-in function with the same name, the UDF becomes inaccessible. To correct this, use `DROP FUNCTION` to drop the UDF, and then use `CREATE FUNCTION` to re-create the UDF with a different nonconflicting name. The same is true if the new version of MySQL implements a built-in function with the same name as an existing stored function. See [Function Name Parsing and Resolution](#), for the rules describing how the server interprets references to different kinds of functions.

10.1.1 Changes Affecting Upgrades to MySQL 5.5

Before upgrading to MySQL 5.5, review the changes described in this section to identify upgrade issues that apply to your current MySQL installation and applications.

Note

In addition to the changes outlined in this section, review the [Release Notes](#) and other important information outlined in [Before You Begin](#).

Changes marked as either **Known issue** or **Incompatible change** are incompatibilities with earlier versions of MySQL, and may require your attention *before you upgrade*. Our aim is to avoid these changes, but occasionally they are necessary to correct problems that would be worse than an incompatibility between releases. If any upgrade issue applicable to your installation involves an incompatibility that requires special handling, follow the instructions given in the incompatibility description.

Sometimes this involves dumping and reloading tables, or use of a statement such as `CHECK TABLE` or `REPAIR TABLE`.

For dump and reload instructions, see [Section 10.3, “Rebuilding or Repairing Tables or Indexes”](#). Any procedure that involves `REPAIR TABLE` with the `USE_FRM` option *must* be done before upgrading. Use of this statement with a version of MySQL different from the one used to create the table (that is, using it after upgrading) may damage the table. See [REPAIR TABLE Syntax](#).

- [Configuration Changes](#)
- [Server Changes](#)
- [SQL Changes](#)

Configuration Changes

- **Incompatible change:** The `InnoDB Plugin` is included in MySQL 5.5 releases. It becomes the built-in version of `InnoDB` in MySQL Server, replacing the version previously included as the built-in `InnoDB` engine. `InnoDB Plugin` is also available in MySQL 5.1 as of 5.1.38, but it is an optional storage engine that must be enabled explicitly using two server options:

```
[mysqld]
ignore-builtin-innodb
plugin-load=innodb=ha_innodb_plugin.so
```

If you were using `InnoDB Plugin` in MySQL 5.1 by means of those options, you must remove them after an upgrade to 5.5 or the server will fail to start.

In addition, in `InnoDB Plugin`, the `innodb_file_io_threads` system variable has been removed and replaced with `innodb_read_io_threads` and `innodb_write_io_threads`. If you upgrade from MySQL 5.1 to MySQL 5.5 and previously explicitly set `innodb_file_io_threads` at server startup, you must change your configuration. Either remove any reference to `innodb_file_io_threads` or replace it with references to `innodb_read_io_threads` and `innodb_write_io_threads`.

- **Incompatible change:** In MySQL 5.5, the server includes a plugin services interface that complements the plugin API. The services interface enables server functionality to be exposed as a “service” that plugins can access through a function-call interface. The `libmysqlservices` library provides access to the available services and dynamic plugins now must be linked against this library (use the `-lmysqlservices` flag). For an example showing how to configure for `CMake`, see [MySQL Services for Plugins](#).

Server Changes

- On Linux systems, the `libaio` library may be needed. Install it first, if it is not already present on your system.
- **Known issue:** As of MySQL 5.5.32, for new installations, the `url` columns in the `mysql` database help tables are now created as type `TEXT` to accommodate longer URLs. For upgrades, `mysql_upgrade` does *not* update the columns. Modify them manually using these statements:

```
ALTER TABLE mysql.help_category MODIFY url TEXT NOT NULL;
ALTER TABLE mysql.help_topic MODIFY url TEXT NOT NULL;
```

- **Incompatible change:** As of MySQL 5.5.3, due to work done for Bug #989, `FLUSH TABLES` is not permitted when there is an active `LOCK TABLES ... READ`. To provide a workaround for this restriction, `FLUSH TABLES` has a new variant, `FLUSH TABLES tbl_list WITH READ LOCK`, that

enables tables to be flushed and locked in a single operation. As a result of this change, applications that previously used this statement sequence to lock and flush tables will fail:

```
LOCK TABLES tbl_list READ;
FLUSH TABLES tbl_list;
```

Such applications should now use this statement instead:

```
FLUSH TABLES tbl_list WITH READ LOCK;
```

- **Incompatible change:** As of MySQL 5.5.7, the server requires that a new grant table, `proxies_priv`, be present in the `mysql` database. If you are upgrading to 5.5.7 from a previous MySQL release rather than performing a new installation, the server will find that this table is missing and exit during startup with the following message:

```
Table 'mysql.proxies_priv' doesn't exist
```

To create the `proxies_priv` table, start the server with the `--skip-grant-tables` option to cause it to skip the normal grant table checks, then run `mysql_upgrade`. For example:

```
mysqld --skip-grant-tables &
mysql_upgrade
```

Then stop the server and restart it normally.

You can specify other options on the `mysqld` command line if necessary. Alternatively, if your installation is configured so that the server normally reads options from an option file, use the `--defaults-file` option to specify the file (enter each command on a single line):

```
mysqld --defaults-file=/usr/local/mysql/etc/my.cnf
--skip-grant-tables &
mysql_upgrade
```

With the `--skip-grant-tables` option, the server does no password or privilege checking, so any client can connect and effectively have all privileges. For additional security, use the `--skip-networking` option as well to prevent remote clients from connecting.

Note

This problem is fixed in MySQL 5.5.8; the server treats a missing `proxies_priv` table as equivalent to an empty table. However, after starting the server, you should still run `mysql_upgrade` to create the table.

- **Incompatible change:** As of MySQL 5.5.7, InnoDB always uses the fast truncation technique, equivalent to `DROP TABLE` and `CREATE TABLE`. It no longer performs a row-by-row delete for tables with parent-child foreign key relationships. `TRUNCATE TABLE` returns an error for such tables. Modify your SQL to issue `DELETE FROM table_name` for such tables instead.
- **Incompatible change:** Prior to MySQL 5.5.7, if you flushed the logs using `FLUSH LOGS` or `mysqladmin flush-logs` and `mysqld` was writing the error log to a file (for example, if it was started with the `--log-error` option), it renames the current log file with the suffix `-old`, then created a new empty log file. This had the problem that a second log-flushing operation thus caused the original error log file to be lost unless you saved it under a different name. For example, you could use the following commands to save the file:

```
mysqladmin flush-logs
mv host_name.err-old backup-directory
```

To avoid the preceding file-loss problem, no renaming occurs as of MySQL 5.5.7; the server merely closes and reopens the log file. To rename the file, you can do so manually before flushing. Then flushing the logs reopens a new file with the original file name. For example, you can rename the file and create a new one using the following commands:

```
mv host_name.err host_name.err-old
mysqladmin flush-logs
mv host_name.err-old backup-directory
```

- **Incompatible change:** As of MySQL 5.5.6, handling of `CREATE TABLE IF NOT EXISTS ... SELECT` statements has been changed for the case that the destination table already exists:
 - Previously, for `CREATE TABLE IF NOT EXISTS ... SELECT`, MySQL produced a warning that the table exists, but inserted the rows and wrote the statement to the binary log anyway. By contrast, `CREATE TABLE ... SELECT` (without `IF NOT EXISTS`) failed with an error, but MySQL inserted no rows and did not write the statement to the binary log.
 - MySQL now handles both statements the same way when the destination table exists, in that neither statement inserts rows or is written to the binary log. The difference between them is that MySQL produces a warning when `IF NOT EXISTS` is present and an error when it is not.

This change in handling of `IF NOT EXISTS` results in an incompatibility for statement-based replication from a MySQL 5.1 master with the original behavior and a MySQL 5.5 slave with the new behavior. Suppose that `CREATE TABLE IF NOT EXISTS ... SELECT` is executed on the master and the destination table exists. The result is that rows are inserted on the master but not on the slave. (Row-based replication does not have this problem.)

To address this issue, statement-based binary logging for `CREATE TABLE IF NOT EXISTS ... SELECT` is changed in MySQL 5.1 as of 5.1.51:

- If the destination table does not exist, there is no change: The statement is logged as is.
- If the destination table does exist, the statement is logged as the equivalent pair of `CREATE TABLE IF NOT EXISTS` and `INSERT ... SELECT` statements. (If the `SELECT` in the original statement is preceded by `IGNORE` or `REPLACE`, the `INSERT` becomes `INSERT IGNORE` or `REPLACE`, respectively.)

This change provides forward compatibility for statement-based replication from MySQL 5.1 to 5.5 because when the destination table exists, the rows will be inserted on both the master and slave. To take advantage of this compatibility measure, the 5.1 server must be at least 5.1.51 and the 5.5 server must be at least 5.5.6.

To upgrade an existing 5.1-to-5.5 replication scenario, upgrade the master first to 5.1.51 or higher. Note that this differs from the usual replication upgrade advice of upgrading the slave first.

A workaround for applications that wish to achieve the original effect (rows inserted regardless of whether the destination table exists) is to use `CREATE TABLE IF NOT EXISTS` and `INSERT ... SELECT` statements rather than `CREATE TABLE IF NOT EXISTS ... SELECT` statements.

Along with the change just described, the following related change was made: Previously, if an existing view was named as the destination table for `CREATE TABLE IF NOT EXISTS ... SELECT`, rows

were inserted into the underlying base table and the statement was written to the binary log. As of MySQL 5.1.51 and 5.5.6, nothing is inserted or logged.

- **Incompatible change:** Prior to MySQL 5.5.6, if the server was started with `character_set_server` set to `utf16`, it crashed during full-text stopword initialization. Now the stopword file is loaded and searched using `latin1` if `character_set_server` is `ucs2`, `utf16`, or `utf32`. If any table was created with `FULLTEXT` indexes while the server character set was `ucs2`, `utf16`, or `utf32`, it should be repaired using this statement:

```
REPAIR TABLE tbl_name QUICK;
```

- **Incompatible change:** As of MySQL 5.5.5, all numeric operators and functions on integer, floating-point and `DECIMAL` values throw an “out of range” error (`ER_DATA_OUT_OF_RANGE`) rather than returning an incorrect value or `NULL`, when the result is out of the supported range for the corresponding data type. See [Out-of-Range and Overflow Handling](#).
- **Incompatible change:** In very old versions of MySQL (prior to 4.1), the `TIMESTAMP` data type supported a display width, which was silently ignored beginning with MySQL 4.1. This is deprecated in MySQL 5.1, and removed altogether in MySQL 5.5. These changes in behavior can lead to two problem scenarios when trying to use `TIMESTAMP(N)` columns with a MySQL 5.5 or later server:
 - When importing a dump file (for example, one created using `mysqldump`) created in a MySQL 5.0 or earlier server into a server from a newer release series, a `CREATE TABLE` or `ALTER TABLE` statement containing `TIMESTAMP(N)` causes the import to fail with a syntax error.

To fix this problem, edit the dump file in a text editor to replace any instances of `TIMESTAMP(N)` with `TIMESTAMP` prior to importing the file. Be sure to use a plain text editor for this, and not a word processor; otherwise, the result is almost certain to be unusable for importing into the MySQL server.

- When trying replicate any `CREATE TABLE` or `ALTER TABLE` statement containing `TIMESTAMP(N)` from a master MySQL server that supports the `TIMESTAMP(N)` syntax to a MySQL 5.5.3 or higher slave, the statement causes replication to fail. Similarly, when you try to restore from a binary log written by a server that supports `TIMESTAMP(N)` to a MySQL 5.5.3 or higher server, any `CREATE TABLE` or `ALTER TABLE` statement containing `TIMESTAMP(N)` causes the backup to fail. This holds true regardless of the logging format.

It may be possible to fix such issues using a hex editor, by replacing any width arguments used with `TIMESTAMP`, and the parentheses containing them, with space characters (hexadecimal `20`). Be sure to use a programmer's binary hex editor and not a regular text editor or word processor for this; otherwise, the result is almost certain to be a corrupted binary log file. To guard against accidental corruption of the binary log, you should always work on a copy of the file rather than the original.

You should try to handle potential issues of these types proactively by updating with `ALTER TABLE` any `TIMESTAMP(N)` columns in your databases so that they use `TIMESTAMP` instead, before performing any upgrades.

- **Incompatible change:** As of MySQL 5.5.3, the Unicode implementation has been extended to provide support for supplementary characters that lie outside the Basic Multilingual Plane (BMP). Noteworthy features:
 - `utf16` and `utf32` character sets have been added. These correspond to the UTF-16 and UTF-32 encodings of the Unicode character set, and they both support supplementary characters.
 - The `utf8mb4` character set has been added. This is similar to `utf8`, but its encoding allows up to four bytes per character to enable support for supplementary characters.

- The `ucs2` character set is essentially unchanged except for the inclusion of some newer BMP characters.

In most respects, upgrading to MySQL 5.5 should present few problems with regard to Unicode usage, although there are some potential areas of incompatibility. These are the primary areas of concern:

- For the variable-length character data types (`VARCHAR` and the `TEXT` types), the maximum length in characters is less for `utf8mb4` columns than for `utf8` columns.
- For all character data types (`CHAR`, `VARCHAR`, and the `TEXT` types), the maximum number of characters that can be indexed is less for `utf8mb4` columns than for `utf8` columns.

Consequently, if you want to upgrade tables from `utf8` to `utf8mb4` to take advantage of supplementary-character support, it may be necessary to change some column or index definitions.

For additional details about the new Unicode character sets and potential incompatibilities, see [Unicode Support](#), and [Converting Between 3-Byte and 4-Byte Unicode Character Sets](#).

- **Incompatible change:** As of MySQL 5.5.3, the server includes `dtoa`, a library for conversion between strings and numbers by David M. Gay. In MySQL, this library provides the basis for improved conversion between string or `DECIMAL` values and approximate-value (`FLOAT` or `DOUBLE`) numbers.

Because the conversions produced by this library differ in some cases from previous results, the potential exists for incompatibilities in applications that rely on previous results. For example, applications that depend on a specific exact result from previous conversions might need adjustment to accommodate additional precision.

For additional information about the properties of `dtoa` conversions, see [Type Conversion in Expression Evaluation](#).

- **Incompatible change:** In MySQL 5.5, several changes were made regarding the language and character set of error messages:
 - The `--language` option for specifying the directory for the error message file is now deprecated. The new `lc_messages_dir` and `lc_messages` system variables should be used instead, and the server treats `--language` as an alias for `lc_messages_dir`.
 - The `language` system variable has been removed and replaced with the new `lc_messages_dir` and `lc_messages` system variables. `lc_messages_dir` has only a global value and is read only. `lc_messages` has global and session values and can be modified at runtime, so the error message language can be changed while the server is running, and individual clients each can have a different error message language by changing their session `lc_messages` value to a different locale name.
 - Error messages previously were constructed in a mix of character sets. This issue is resolved by constructing error messages internally within the server using UTF-8 and returning them to the client in the character set specified by the `character_set_results` system variable. The content of error messages therefore may in some cases differ from the messages returned previously.

For more information, see [Setting the Error Message Language](#), and [Error Message Character Set](#).

- **Incompatible change:** MySQL 5.5 implements new functions used to calculate row placement for tables partitioned by `KEY` and `LINEAR KEY`. Tables that were created using `KEY` or `LINEAR KEY` partitioning in MySQL 5.1 can be upgraded in MySQL 5.5.31 and later using `ALTER TABLE ... PARTITION BY ALGORITHM=2 [LINEAR] KEY (...)`. (Bug #14521864, Bug #66462)

SQL Changes

- **Incompatible change:** Previously, the parser accepted an `INTO` clause in nested `SELECT` statements, which is invalid because such statements must return their results to the outer context. As of MySQL 5.5.3, this syntax is no longer permitted and statements that use it must be changed.
- **Incompatible change:** In MySQL 5.5.3, several changes were made to alias resolution in multiple-table `DELETE` statements so that it is no longer possible to have inconsistent or ambiguous table aliases.
- In MySQL 5.1.23, alias declarations outside the `table_references` part of the statement were disallowed for the `USING` variant of multiple-table `DELETE` syntax, to reduce the possibility of ambiguous aliases that could lead to ambiguous statements that have unexpected results such as deleting rows from the wrong table.

As of MySQL 5.5.3, alias declarations outside `table_references` are disallowed for all multiple-table `DELETE` statements. Alias declarations are permitted only in the `table_references` part.

Incorrect:

```
DELETE FROM t1 AS a2 USING t1 AS a1 INNER JOIN t2 AS a2;
DELETE t1 AS a2 FROM t1 AS a1 INNER JOIN t2 AS a2;
```

Correct:

```
DELETE FROM t1 USING t1 AS a1 INNER JOIN t2 AS a2;
DELETE t1 FROM t1 AS a1 INNER JOIN t2 AS a2;
```

- Previously, for alias references in the list of tables from which to delete rows in a multiple-table delete, the default database is used unless one is specified explicitly. For example, if the default database is `db1`, the following statement does not work because the unqualified alias reference `a2` is interpreted as having a database of `db1`:

```
DELETE a1, a2 FROM db1.t1 AS a1 INNER JOIN db2.t2 AS a2
WHERE a1.id=a2.id;
```

To correctly match an alias that refers to a table outside the default database, you must explicitly qualify the reference with the name of the proper database:

```
DELETE a1, db2.a2 FROM db1.t1 AS a1 INNER JOIN db2.t2 AS a2
WHERE a1.id=a2.id;
```

As of MySQL 5.5.3, alias resolution does not require qualification and alias references should not be qualified with the database name. Qualified names are interpreted as referring to tables, not aliases.

Statements containing alias constructs that are no longer permitted must be rewritten.

- Some keywords may be reserved in MySQL 5.5 that were not reserved in MySQL 5.1. See [Keywords and Reserved Words](#). This can cause words previously used as identifiers to become illegal. To fix affected statements, use identifier quoting. See [Schema Object Names](#).

10.2 Downgrading MySQL

This section describes how to downgrade to an older MySQL version.

- [Downgrade Methods](#)

- [Downgrade Paths](#)
- [Before You Begin](#)
- [Performing an In-Place Downgrade](#)
- [Performing a Logical Downgrade](#)
- [Downgrade Troubleshooting](#)

Note

In the following discussion, MySQL commands that must be run using a MySQL account with administrative privileges include `-u root` on the command line to specify the MySQL `root` user. Commands that require a password for `root` also include a `-p` option. Because `-p` is followed by no option value, such commands prompt for the password. Type the password when prompted and press Enter.

SQL statements can be executed using the `mysql` command-line client (connect as `root` to ensure that you have the necessary privileges).

Downgrade Methods

Supported downgrade methods include:

- *In-Place Downgrade*: Involves shutting down the new MySQL version, replacing the new MySQL binaries or packages with the old ones, and restarting the old MySQL version on the new data files. In-place downgrades are supported for downgrades between GA versions within the same release series. For example, in-place downgrades are supported for downgrades from 5.5.46 to 5.5.45.
- *Logical Downgrade*: Involves using `mysqldump` to dump all tables from the new MySQL version, and then loading the dump file into the old MySQL version. Logical downgrades are supported for downgrades between GA versions within the same release series and for downgrades between release levels. For example, logical downgrades are supported for downgrades from 5.5.46 to 5.5.45 and for downgrades from 5.5 to 5.1.

For procedures, see [Performing an In-Place Downgrade](#), and [Performing a Logical Downgrade](#).

Downgrade Paths

- Downgrade is only supported between General Availability (GA) releases.
- Downgrade from MySQL 5.5 to 5.1 is supported using the *logical downgrade* method.
- Downgrade that skips versions is not supported. For example, downgrading directly from MySQL 5.5 to 5.0 is not supported.
- Downgrade within a release series is supported. For example, downgrading from MySQL 5.5.z to 5.5.y is supported. Skipping a release is also supported. For example, downgrading from MySQL 5.5.z to 5.5.x is supported.

Before You Begin

Before downgrading, the following steps are recommended:

- Review the [Release Notes](#) for the MySQL version you are downgrading from to ensure that there are no features or fixes that you really need.

- Review [Section 10.2.1, “Changes Affecting Downgrades from MySQL 5.5”](#). This section describes changes that may require action before or after downgrading.

Note

The downgrade procedures described in the following sections assume you are downgrading with data files created or modified by the newer MySQL version. However, if you did not modify your data after upgrading, downgrading using backups taken *before* upgrading to the new MySQL version is recommended. Many of the changes described in [Section 10.2.1, “Changes Affecting Downgrades from MySQL 5.5”](#) that require action before or after downgrading are not applicable when downgrading using backups taken *before* upgrading to the new MySQL version.

- Always back up your current databases and log files before downgrading. The backup should include the `mysql` database, which contains the MySQL system tables. See [Database Backup Methods](#).
- Use of new features, new configuration options, or new configuration option values that are not supported by a previous release may cause downgrade errors or failures. Before downgrading, it is recommended that you reverse changes resulting from the use of new features and remove configuration settings that are not supported by the release you are downgrading to.
- If you use XA transactions with `InnoDB`, run `XA RECOVER` before downgrading to check for uncommitted XA transactions. If results are returned, either commit or rollback the XA transactions by issuing an `XA COMMIT` or `XA ROLLBACK` statement.

Performing an In-Place Downgrade

In-place downgrades are supported for downgrades between GA releases within the same release series. Before proceeding, review [Before You Begin](#).

To perform an in-place downgrade:

1. Review the changes described in [Section 10.2.1, “Changes Affecting Downgrades from MySQL 5.5”](#) for steps to be performed before downgrading.
2. If you use `InnoDB`, configure MySQL to perform a slow shutdown by setting `innodb_fast_shutdown` to 0. For example:

```
mysql -u root -p --execute="SET GLOBAL innodb_fast_shutdown=0"
```

With a slow shutdown, `InnoDB` performs a full purge and change buffer merge before shutting down, which ensures that data files are fully prepared in case of file format differences between releases.

3. Shut down the newer MySQL server. For example:

```
mysqladmin -u root -p shutdown
```

4. After the slow shutdown, remove the `InnoDB` redo log files (the `ib_logfile*` files) from the `data` directory to avoid downgrade issues related to redo log file format changes that may have occurred between releases.

```
rm ib_logfile*
```

5. Downgrade the MySQL binaries or packages in-place by replacing the newer binaries or packages with the older ones.

6. Start the older (downgraded) MySQL server, using the existing data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir
```

7. Run `mysql_upgrade`. For example:

```
mysql_upgrade -u root -p
```

8. Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown  
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir
```

Performing a Logical Downgrade

Logical downgrades are supported for downgrades between releases within the same release series and for downgrades to the previous release level. Only downgrades between General Availability (GA) releases are supported. Before proceeding, review [Before You Begin](#).

To perform a logical downgrade:

1. Review the changes described in [Section 10.2.1, “Changes Affecting Downgrades from MySQL 5.5”](#) for steps to be performed before downgrading.
2. Dump all databases. For example:

```
mysqldump -u root -p  
--add-drop-table --routines --events  
--all-databases --force > data-for-downgrade.sql
```

3. Shut down the newer MySQL server. For example:

```
mysqladmin -u root -p shutdown
```

4. Initialize an older MySQL instance, with a new data directory. For example:

```
scripts/mysql_install_db --user=mysql
```

5. Start the older MySQL server, using the new data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/new-datadir
```

6. Load the dump file into the older MySQL server. For example:

```
mysql -u root -p --force < data-for-upgrade.sql
```

7. Run `mysql_upgrade`. For example:

```
mysql_upgrade -u root -p
```

8. Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/new-datadir
```

Downgrade Troubleshooting

If you downgrade from one release series to another, there may be incompatibilities in table storage formats. In this case, use `mysqldump` to dump your tables before downgrading. After downgrading, reload the dump file using `mysql` or `mysqlimport` to re-create your tables. For examples, see [Section 10.4, “Copying MySQL Databases to Another Machine”](#).

A typical symptom of a downward-incompatible table format change when you downgrade is that you cannot open tables. In that case, use the following procedure:

1. Stop the older MySQL server that you are downgrading to.
2. Restart the newer MySQL server you are downgrading from.
3. Dump any tables that were inaccessible to the older server by using `mysqldump` to create a dump file.
4. Stop the newer MySQL server and restart the older one.
5. Reload the dump file into the older server. Your tables should be accessible.

10.2.1 Changes Affecting Downgrades from MySQL 5.5

Before downgrading from MySQL 5.5, review the changes described in this section. Some changes may require action before or after downgrading.

- **System Tables.** The `mysql.proc.comment` column definition changed between MySQL 5.1 and 5.5. After downgrading from 5.5 to 5.1, this table is seen as corrupt and in need of repair. Running `mysql_upgrade` from the version of MySQL to which you downgraded (as documented in the downgrade procedures) reverts the `mysql.proc.comment` column definition.
- **InnoDB.** MySQL 5.5 uses [InnoDB Plugin](#) as the built-in version of [InnoDB](#). MySQL 5.1 includes [InnoDB Plugin](#) as of 5.1.38, but as an option that must be enabled explicitly. See the [Release Notes](#) for MySQL 5.1.38.
- **InnoDB.** In MySQL 5.5.14, the length limit for index prefix keys is increased from 767 bytes to 3072 bytes, for [InnoDB](#) tables using `ROW_FORMAT=DYNAMIC` or `ROW_FORMAT=COMPRESSED`. See [Limits on InnoDB Tables](#) for details. If you downgrade from one of these releases or higher, to an earlier release with a lower length limit, the index prefix keys could be truncated at 767 bytes or the downgrade could fail. This issue could only occur if the configuration option `innodb_large_prefix` was enabled on the server being downgraded.
- **Tables partitioned by [LINEAR] KEY.** MySQL 5.5 implements new functions used to calculate row placement for tables partitioned by `KEY` and `LINEAR KEY`. Tables that were created using `KEY` or `LINEAR KEY` partitioning in MySQL 5.5 cannot be used by a MySQL 5.1 server. In MySQL 5.5.31 and later, you can downgrade such tables with `ALTER TABLE ... PARTITION BY ALGORITHM=1 [LINEAR] KEY (...)` to make them compatible with MySQL 5.1.

10.2.2 Downgrading from MySQL Enterprise Edition to MySQL Community Server

This section describes the steps required to downgrade from MySQL Enterprise Edition to MySQL Community Edition. This can be done at any time, and is required at the expiration of a MySQL Enterprise Edition subscription if you wish to continue using MySQL Server.

When you perform such a downgrade, all commercially licensed components of the MySQL Enterprise Edition subscription must be uninstalled. These components and related considerations are described in the rest of this section.

Note

The issues described in this section are in addition to any that may be encountered as a result of any upgrade or downgrade of the MySQL Server version (such as between MySQL 5.5 and 5.1). Information about upgrading and downgrading between MySQL release series can be found elsewhere in this chapter; see [Section 10.1, “Upgrading MySQL”](#), and [Section 10.2, “Downgrading MySQL”](#).

MySQL Enterprise Database Server. All commercial versions of MySQL Database Server must be uninstalled.

Commercially licensed extensions. All commercially licensed MySQL Enterprise Database Server extensions must be uninstalled. This includes the following commercial extensions:

- *MySQL External Authentication for Windows:* Following uninstallation of this plugin, existing MySQL user accounts must be re-created using local authentication. See [MySQL User Account Management](#), for more information.
- *MySQL External Authentication for PAM:* Following uninstallation of this plugin, existing MySQL user accounts must be re-created using local authentication. See [MySQL User Account Management](#), for more information.
- *MySQL Enterprise Thread Pool:* Following uninstallation of this plugin, existing MySQL servers revert to default thread and connection handling.
- *MySQL Enterprise Audit:* Following uninstallation of this plugin, no logging of user logins or query activity occurs.
- *MySQL High Availability:* Following uninstallation of this plugin, automated failover is no longer available.

MySQL Enterprise Backup. MySQL Enterprise Backup must be uninstalled. Uninstalling this application has the effects listed here:

- Automated backup scripts no longer work.
- Existing backup images taken with MySQL Enterprise Backup can no longer be used for recovery.
- Third-party integration with multimedia systems such as NetBackup, Tivoli, and Oracle Secure Backup no longer works.

MySQL Enterprise Monitor, MySQL Query Analyzer, agents. MySQL Enterprise Monitor, MySQL Query Analyzer, and all server-side agents must be uninstalled. Uninstalling these applications and agents has the following effects:

- Automated SNMP and SMTP alerts no longer work.
- All historical MySQL, OS monitoring, query, and performance metrics as well as all trending data are lost.
- All environment-specific monitoring templates, custom advisors, graphs and scripts are also lost.

10.3 Rebuilding or Repairing Tables or Indexes

This section describes how to rebuild or repair tables or indexes, which may be necessitated by:

- Changes to how MySQL handles data types or character sets. For example, an error in a collation might have been corrected, necessitating a table rebuild to update the indexes for character columns that use the collation.
- Required table repairs or upgrades reported by `CHECK TABLE`, `mysqlcheck`, or `mysql_upgrade`.

Methods for rebuilding a table include:

- [Dump and Reload Method](#)
- [ALTER TABLE Method](#)
- [REPAIR TABLE Method](#)

Dump and Reload Method

If you are rebuilding tables because a different version of MySQL will not handle them after a binary (in-place) upgrade or downgrade, you must use the dump-and-reload method. Dump the tables *before* upgrading or downgrading using your original version of MySQL. Then reload the tables *after* upgrading or downgrading.

If you use the dump-and-reload method of rebuilding tables only for the purpose of rebuilding indexes, you can perform the dump either before or after upgrading or downgrading. Reloading still must be done afterward.

If you need to rebuild an `InnoDB` table because a `CHECK TABLE` operation indicates that a table upgrade is required, use `mysqldump` to create a dump file and `mysql` to reload the file. If the `CHECK TABLE` operation indicates that there is a corruption or causes `InnoDB` to fail, refer to [Forcing InnoDB Recovery](#) for information about using the `innodb_force_recovery` option to restart `InnoDB`. To understand the type of problem that `CHECK TABLE` may be encountering, refer to the `InnoDB` notes in [CHECK TABLE Syntax](#).

To rebuild a table by dumping and reloading it, use `mysqldump` to create a dump file and `mysql` to reload the file:

```
mysqldump db_name t1 > dump.sql
mysql db_name < dump.sql
```

To rebuild all the tables in a single database, specify the database name without any following table name:

```
mysqldump db_name > dump.sql
mysql db_name < dump.sql
```

To rebuild all tables in all databases, use the `--all-databases` option:

```
mysqldump --all-databases > dump.sql
mysql < dump.sql
```

ALTER TABLE Method

To rebuild a table with `ALTER TABLE`, use a “null” alteration; that is, an `ALTER TABLE` statement that “changes” the table to use the storage engine that it already has. For example, if `t1` is an `InnoDB` table, use this statement:

```
ALTER TABLE t1 ENGINE = InnoDB;
```

If you are not sure which storage engine to specify in the `ALTER TABLE` statement, use `SHOW CREATE TABLE` to display the table definition.

REPAIR TABLE Method

The `REPAIR TABLE` method is only applicable to `MyISAM`, `ARCHIVE`, and `CSV` tables.

You can use `REPAIR TABLE` if the table checking operation indicates that there is a corruption or that an upgrade is required. For example, to repair a `MyISAM` table, use this statement:

```
REPAIR TABLE t1;
```

`mysqlcheck --repair` provides command-line access to the `REPAIR TABLE` statement. This can be a more convenient means of repairing tables because you can use the `--databases` or `--all-databases` option to repair all tables in specific databases or all databases, respectively:

```
mysqlcheck --repair --databases db_name ...  
mysqlcheck --repair --all-databases
```

10.4 Copying MySQL Databases to Another Machine

In cases where you need to transfer databases between different architectures, you can use `mysqldump` to create a file containing SQL statements. You can then transfer the file to the other machine and feed it as input to the `mysql` client.

Note

You can copy the `.frm`, `.MYI`, and `.MYD` files for `MyISAM` tables between different architectures that support the same floating-point format. (MySQL takes care of any byte-swapping issues.) See [The MyISAM Storage Engine](#).

Use `mysqldump --help` to see what options are available.

The easiest (although not the fastest) way to move a database between two machines is to run the following commands on the machine on which the database is located:

```
mysqladmin -h 'other_hostname' create db_name  
mysqldump db_name | mysql -h 'other_hostname' db_name
```

If you want to copy a database from a remote machine over a slow network, you can use these commands:

```
mysqladmin create db_name  
mysqldump -h 'other_hostname' --compress db_name | mysql db_name
```

You can also store the dump in a file, transfer the file to the target machine, and then load the file into the database there. For example, you can dump a database to a compressed file on the source machine like this:

```
mysqldump --quick db_name | gzip > db_name.gz
```

Transfer the file containing the database contents to the target machine and run these commands there:

```
mysqladmin create db_name
gunzip < db_name.gz | mysql db_name
```

You can also use `mysqldump` and `mysqlimport` to transfer the database. For large tables, this is much faster than simply using `mysqldump`. In the following commands, `DUMPDIR` represents the full path name of the directory you use to store the output from `mysqldump`.

First, create the directory for the output files and dump the database:

```
mkdir DUMPDIR
mysqldump --tab=DUMPDIR db_name
```

Then transfer the files in the `DUMPDIR` directory to some corresponding directory on the target machine and load the files into MySQL there:

```
mysqladmin create db_name # create database
cat DUMPDIR/*.sql | mysql db_name # create tables in database
mysqlimport db_name DUMPDIR/*.txt # load data into tables
```

Do not forget to copy the `mysql` database because that is where the grant tables are stored. You might have to run commands as the MySQL `root` user on the new machine until you have the `mysql` database in place.

After you import the `mysql` database on the new machine, execute `mysqladmin flush-privileges` so that the server reloads the grant table information.

Chapter 11 MySQL Program Environment Variables

This section lists environment variables that are used directly or indirectly by MySQL. Most of these can also be found in other places in this manual.

Options on the command line take precedence over values specified in option files and environment variables, and values in option files take precedence over values in environment variables. In many cases, it is preferable to use an option file instead of environment variables to modify the behavior of MySQL. See [Using Option Files](#).

Variable	Description
CXX	The name of your C++ compiler (for running <code>CMake</code>).
CC	The name of your C compiler (for running <code>CMake</code>).
DBI_USER	The default user name for Perl DBI.
DBI_TRACE	Trace options for Perl DBI.
HOME	The default path for the <code>mysql</code> history file is <code>\$HOME/.mysql_history</code> .
LD_RUN_PATH	Used to specify the location of <code>libmysqlclient.so</code> .
LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN	Enable <code>mysql_clear_password</code> authentication plugin; see Client-Side Cleartext Pluggable Authentication .
MYSQL_DEBUG	Debug trace options when debugging.
MYSQL_GROUP_SUFFIX	Option group suffix value (like specifying <code>--defaults-group-suffix</code>).
MYSQL_HISTFILE	The path to the <code>mysql</code> history file. If this variable is set, its value overrides the default for <code>\$HOME/.mysql_history</code> .
MYSQL_HOME	The path to the directory in which the server-specific <code>my.cnf</code> file resides.
MYSQL_HOST	The default host name used by the <code>mysql</code> command-line client.
MYSQL_PS1	The command prompt to use in the <code>mysql</code> command-line client.
MYSQL_PWD	The default password when connecting to <code>mysqld</code> . Using this is insecure. See End-User Guidelines for Password Security .
MYSQL_TCP_PORT	The default TCP/IP port number.
MYSQL_UNIX_PORT	The default Unix socket file name; used for connections to <code>localhost</code> .
PATH	Used by the shell to find MySQL programs.
TMPDIR	The directory in which temporary files are created.
TZ	This should be set to your local time zone. See Time Zone Problems .
UMASK	The user-file creation mode when creating files. See note following table.
UMASK_DIR	The user-directory creation mode when creating directories. See note following table.

Variable	Description
<code>USER</code>	The default user name on Windows when connecting to <code>mysqld</code> .

For information about the `mysql` history file, see [mysql Logging](#).

The default `UMASK` and `UMASK_DIR` values are `0660` and `0700`, respectively. MySQL assumes that the value for `UMASK` or `UMASK_DIR` is in octal if it starts with a zero. For example, setting `UMASK=0600` is equivalent to `UMASK=384` because `0600` octal is `384` decimal.

The `UMASK` and `UMASK_DIR` variables, despite their names, are used as modes, not masks:

- If `UMASK` is set, `mysqld` uses $(\$UMASK | 0600)$ as the mode for file creation, so that newly created files have a mode in the range from `0600` to `0666` (all values octal).
- If `UMASK_DIR` is set, `mysqld` uses $(\$UMASK_DIR | 0700)$ as the base mode for directory creation, which then is AND-ed with $\sim(\sim\$UMASK \& 0666)$, so that newly created directories have a mode in the range from `0700` to `0777` (all values octal). The AND operation may remove read and write permissions from the directory mode, but not execute permissions.

Chapter 12 Perl Installation Notes

Table of Contents

12.1 Installing Perl on Unix	189
12.2 Installing ActiveState Perl on Windows	190
12.3 Problems Using the Perl DBI/DBD Interface	191

The Perl `DBI` module provides a generic interface for database access. You can write a `DBI` script that works with many different database engines without change. To use `DBI`, you must install the `DBI` module, as well as a DataBase Driver (DBD) module for each type of database server you want to access. For MySQL, this driver is the `DBD::mysql` module.

Perl, and the `DBD::MySQL` module for `DBI` must be installed if you want to run the MySQL benchmark scripts; see [The MySQL Benchmark Suite](#). They are also required for the NDB Cluster `ndb_size.pl` utility; see [ndb_size.pl — NDBCLUSTER Size Requirement Estimator](#).

Note

Perl support is not included with MySQL distributions. You can obtain the necessary modules from <http://search.cpan.org> for Unix, or by using the ActiveState `ppm` program on Windows. The following sections describe how to do this.

The `DBI/DBD` interface requires Perl 5.6.0, and 5.6.1 or later is preferred. `DBI` *does not work* if you have an older version of Perl. You should use `DBD::mysql` 4.009 or higher. Although earlier versions are available, they do not support the full functionality of MySQL 5.5.

12.1 Installing Perl on Unix

MySQL Perl support requires that you have installed MySQL client programming support (libraries and header files). Most installation methods install the necessary files. If you install MySQL from RPM files on Linux, be sure to install the developer RPM as well. The client programs are in the client RPM, but client programming support is in the developer RPM.

The files you need for Perl support can be obtained from the CPAN (Comprehensive Perl Archive Network) at <http://search.cpan.org>.

The easiest way to install Perl modules on Unix is to use the `CPAN` module. For example:

```
shell> perl -MCPAN -e shell
cpan> install DBI
cpan> install DBD::mysql
```

The `DBD::mysql` installation runs a number of tests. These tests attempt to connect to the local MySQL server using the default user name and password. (The default user name is your login name on Unix, and `ODBC` on Windows. The default password is “no password.”) If you cannot connect to the server with those values (for example, if your account has a password), the tests fail. You can use `force install DBD::mysql` to ignore the failed tests.

`DBI` requires the `Data::Dumper` module. It may be installed; if not, you should install it before installing `DBI`.

It is also possible to download the module distributions in the form of compressed `tar` archives and build the modules manually. For example, to unpack and build a DBI distribution, use a procedure such as this:

1. Unpack the distribution into the current directory:

```
shell> gunzip < DBI-VERSION.tar.gz | tar xvf -
```

This command creates a directory named `DBI-VERSION`.

2. Change location into the top-level directory of the unpacked distribution:

```
shell> cd DBI-VERSION
```

3. Build the distribution and compile everything:

```
shell> perl Makefile.PL
shell> make
shell> make test
shell> make install
```

The `make test` command is important because it verifies that the module is working. Note that when you run that command during the `DBD:mysql` installation to exercise the interface code, the MySQL server must be running or the test fails.

It is a good idea to rebuild and reinstall the `DBD:mysql` distribution whenever you install a new release of MySQL. This ensures that the latest versions of the MySQL client libraries are installed correctly.

If you do not have access rights to install Perl modules in the system directory or if you want to install local Perl modules, the following reference may be useful: <http://learn.perl.org/faq/perlfaq8.html#How-do-I-keep-my-own-module-library-directory->

12.2 Installing ActiveState Perl on Windows

On Windows, you should do the following to install the MySQL `DBD` module with ActiveState Perl:

1. Get ActiveState Perl from <http://www.activestate.com/Products/ActivePerl/> and install it.
2. Open a console window.
3. If necessary, set the `HTTP_proxy` variable. For example, you might try a setting like this:

```
C:\> set HTTP_proxy=my.proxy.com:3128
```

4. Start the PPM program:

```
C:\> C:\perl\bin\ppm.pl
```

5. If you have not previously done so, install `DBI`:

```
ppm> install DBI
```

6. If this succeeds, run the following command:

```
ppm> install DBD-mysql
```

This procedure should work with ActiveState Perl 5.6 or higher.

If you cannot get the procedure to work, you should install the ODBC driver instead and connect to the MySQL server through ODBC:

```
use DBI;
$dbh= DBI->connect("DBI:ODBC:$dsn",$user,$password) ||
die "Got error $DBI::errstr when connecting to $dsn\n";
```

12.3 Problems Using the Perl DBI/DBD Interface

If Perl reports that it cannot find the `./mysql/mysql.so` module, the problem is probably that Perl cannot locate the `libmysqlclient.so` shared library. You should be able to fix this problem by one of the following methods:

- Copy `libmysqlclient.so` to the directory where your other shared libraries are located (probably `/usr/lib` or `/lib`).
- Modify the `-L` options used to compile `DBD:mysql` to reflect the actual location of `libmysqlclient.so`.
- On Linux, you can add the path name of the directory where `libmysqlclient.so` is located to the `/etc/ld.so.conf` file.
- Add the path name of the directory where `libmysqlclient.so` is located to the `LD_RUN_PATH` environment variable. Some systems use `LD_LIBRARY_PATH` instead.

Note that you may also need to modify the `-L` options if there are other libraries that the linker fails to find. For example, if the linker cannot find `libc` because it is in `/lib` and the link command specifies `-L/usr/lib`, change the `-L` option to `-L/lib` or add `-L/lib` to the existing link command.

If you get the following errors from `DBD:mysql`, you are probably using `gcc` (or using an old binary compiled with `gcc`):

```
/usr/bin/perl: can't resolve symbol '__moddi3'
/usr/bin/perl: can't resolve symbol '__divdi3'
```

Add `-L/usr/lib/gcc-lib/... -lgcc` to the link command when the `mysql.so` library gets built (check the output from `make` for `mysql.so` when you compile the Perl client). The `-L` option should specify the path name of the directory where `libgcc.a` is located on your system.

Another cause of this problem may be that Perl and MySQL are not both compiled with `gcc`. In this case, you can solve the mismatch by compiling both with `gcc`.

