MySQL and Windows
Abstract

This is the MySQL™ extract for Microsoft Windows from the MySQL 8.0 Reference Manual.

For legal information, see the Legal Notices.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2020-03-25 (revision: 65472)
# Table of Contents

Preface and Legal Notices ........................................................................................................... v
1 Installing MySQL on Microsoft Windows .................................................................................. 1
  1.1 MySQL Installation Layout on Microsoft Windows ......................................................... 4
  1.2 Choosing an Installation Package .................................................................................. 4
  1.3 MySQL Installer for Windows ......................................................................................... 5
    1.3.1 MySQL Installer Initial Setup ........................................................................... 7
    1.3.2 Setting Alternative Server Paths with MySQL Installer .................................. 10
    1.3.3 Installation Workflow with MySQL Installer .................................................. 11
    1.3.4 MySQL Installer Product Catalog and Dashboard .......................................... 21
    1.3.5 MySQLInstallerConsole Reference ................................................................ 27
  1.4 Installing MySQL on Microsoft Windows Using a `noinstall` ZIP Archive ................. 30
    1.4.1 Extracting the Install Archive ........................................................................... 30
    1.4.2 Creating an Option File .................................................................................... 30
    1.4.3 Selecting a MySQL Server Type ...................................................................... 32
    1.4.4 Initializing the Data Directory .......................................................................... 32
    1.4.5 Starting the Server for the First Time .............................................................. 32
    1.4.6 Starting MySQL from the Windows Command Line ........................................ 33
    1.4.7 Customizing the PATH for MySQL Tools ....................................................... 34
    1.4.8 Starting MySQL as a Windows Service ............................................................. 35
    1.4.9 Testing The MySQL Installation ....................................................................... 37
  1.5 Troubleshooting a Microsoft Windows MySQL Server Installation ....................... 38
  1.6 Windows Postinstallation Procedures .............................................................................. 39
  1.7 Windows Platform Restrictions ....................................................................................... 41
2 Upgrading MySQL on Windows .............................................................................................. 45
3 Connection to MySQL Server Failing on Windows ............................................................... 47
4 Resetting the Root Password: Windows Systems ................................................................. 49
5 MySQL Notifier ....................................................................................................................... 51
  5.1 MySQL Notifier Overview .............................................................................................. 51
    5.1.1 MySQL Notifier Tasks ...................................................................................... 51
    5.1.2 MySQL Notifier Options .................................................................................. 52
    5.1.3 Managing Monitored Items .............................................................................. 53
    5.1.4 Troubleshooting ............................................................................................... 59
  5.2 Setting Up Remote Monitoring in MySQL Notifier ...................................................... 59
6 MySQL for Excel ..................................................................................................................... 65
  6.1 Introduction ....................................................................................................................... 65
  6.2 Installation ........................................................................................................................ 66
  6.3 Configuration .................................................................................................................... 67
    6.3.1 MySQL Connections in Excel ........................................................................... 67
    6.3.2 New Schema Creation in Excel ....................................................................... 70
    6.3.3 Global Options .................................................................................................. 71
  6.4 What is New in MySQL for Excel ................................................................................... 77
    6.4.1 What is New in MySQL for Excel 1.3 .............................................................. 77
    6.4.2 What is New in MySQL for Excel 1.2 .............................................................. 79
  6.5 Edit MySQL Data in Excel .............................................................................................. 80
    6.5.1 Edit Session Overview ...................................................................................... 80
    6.5.2 Multiple Edit Sessions ...................................................................................... 82
  6.6 Import MySQL Data into Excel ........................................................................................ 82
    6.6.1 Importing a Procedure ...................................................................................... 84
    6.6.2 Import — Advanced Options ........................................................................... 85
    6.6.3 Adding Summary Fields .................................................................................... 87
    6.6.4 Creating PivotTables ....................................................................................... 88
  6.7 Append Excel Data into MySQL .................................................................................... 97
    6.7.1 Column Mappings ............................................................................................. 97
    6.7.2 Append — Advanced Options .......................................................................... 99
  6.8 Export Excel Data into MySQL ..................................................................................... 102
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 General Information</td>
<td>109</td>
</tr>
<tr>
<td>7.1.1 New in Version 2.0</td>
<td>110</td>
</tr>
<tr>
<td>7.1.2 New in Version 1.2</td>
<td>114</td>
</tr>
<tr>
<td>7.2 Installing MySQL for Visual Studio</td>
<td>116</td>
</tr>
<tr>
<td>7.3 Enabling the MySQL Toolbar</td>
<td>118</td>
</tr>
<tr>
<td>7.4 Making a Connection</td>
<td>119</td>
</tr>
<tr>
<td>7.4.1 Connect Using Server Explorer</td>
<td>121</td>
</tr>
<tr>
<td>7.4.2 Connect Using MySQL Connections Manager</td>
<td>123</td>
</tr>
<tr>
<td>7.5 Editing</td>
<td>125</td>
</tr>
<tr>
<td>7.5.1 MySQL SQL Editor</td>
<td>125</td>
</tr>
<tr>
<td>7.5.2 Code Editors</td>
<td>126</td>
</tr>
<tr>
<td>7.5.3 Editing Tables</td>
<td>128</td>
</tr>
<tr>
<td>7.5.4 Editing Views</td>
<td>133</td>
</tr>
<tr>
<td>7.5.5 Editing Indexes</td>
<td>135</td>
</tr>
<tr>
<td>7.5.6 Editing Foreign Keys</td>
<td>136</td>
</tr>
<tr>
<td>7.5.7 Editing Stored Procedures and Functions</td>
<td>137</td>
</tr>
<tr>
<td>7.5.8 Editing Triggers</td>
<td>139</td>
</tr>
<tr>
<td>7.6 MySQL Project Items</td>
<td>140</td>
</tr>
<tr>
<td>7.6.1 MySQL ASP.NET MVC Items</td>
<td>140</td>
</tr>
<tr>
<td>7.6.2 MySQL Windows Forms Items</td>
<td>147</td>
</tr>
<tr>
<td>7.7 MySQL Application Configuration Tool</td>
<td>149</td>
</tr>
<tr>
<td>7.7.1 Entity Framework</td>
<td>150</td>
</tr>
<tr>
<td>7.7.2 Web Providers</td>
<td>152</td>
</tr>
<tr>
<td>7.7.3 Using the MySQL Connection String Editor</td>
<td>156</td>
</tr>
<tr>
<td>7.8 MySQL Data Export Tool</td>
<td>157</td>
</tr>
<tr>
<td>7.9 DDL T4 Template Macro</td>
<td>163</td>
</tr>
<tr>
<td>7.10 Debugging Stored Procedures and Functions</td>
<td>164</td>
</tr>
<tr>
<td>7.11 MySQL for Visual Studio Frequently Asked Questions</td>
<td>174</td>
</tr>
</tbody>
</table>
Preface and Legal Notices

This is the MySQL extract for Microsoft Windows from the MySQL 8.0 Reference Manual.

Licensing information—MySQL 8.0. This product may include third-party software, used under license. If you are using a Commercial release of MySQL 8.0, see the MySQL 8.0 Commercial Release License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a Community release of MySQL 8.0, see the MySQL 8.0 Community Release License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

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

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

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

Legal Notices

Copyright © 1997, 2020, 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.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.
Chapter 1 Installing MySQL on Microsoft Windows

Table of Contents

1.1 MySQL Installation Layout on Microsoft Windows .......................................................... 4
1.2 Choosing an Installation Package ......................................................................................... 4
1.3 MySQL Installer for Windows .................................................................................................. 5
  1.3.1 MySQL Installer Initial Setup .......................................................................................... 7
  1.3.2 Setting Alternative Server Paths with MySQL Installer ................................................. 10
  1.3.3 Installation Workflow with MySQL Installer ................................................................. 11
  1.3.4 MySQL Installer Product Catalog and Dashboard ......................................................... 21
  1.3.5 MySQLInstallerConsole Reference .............................................................................. 27
1.4 Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive ......................... 30
  1.4.1 Extracting the Install Archive ....................................................................................... 30
  1.4.2 Creating an Option File ................................................................................................. 30
  1.4.3 Selecting a MySQL Server Type .................................................................................... 32
  1.4.4 Initializing the Data Directory ....................................................................................... 32
  1.4.5 Starting the Server for the First Time ........................................................................... 32
  1.4.6 Starting MySQL from the Windows Command Line .................................................... 33
  1.4.7 Customizing the PATH for MySQL Tools ....................................................................... 34
  1.4.8 Starting MySQL as a Windows Service ......................................................................... 35
  1.4.9 Testing The MySQL Installation .................................................................................... 37
1.5 Troubleshooting a Microsoft Windows MySQL Server Installation ................................. 38
1.6 Windows Postinstallation Procedures .................................................................................. 39
1.7 Windows Platform Restrictions ............................................................................................ 41

Important
MySQL 8.0 Server requires the Microsoft Visual C++ 2015 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. Additionally, MySQL debug binaries require Visual Studio 2015 to be installed.

MySQL is available for Microsoft Windows 64-bit operating systems only. For supported Windows platform information, see https://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 a specific version of MySQL Server as follows:


   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. Determine the setup type to use for the initial installation of MySQL products. For example:

   • **Developer Default**: Provides a setup type that includes the selected version of MySQL Server and other MySQL tools related to MySQL development, such as MySQL Workbench.
Additional Installation Information

• **Server Only**: Provides a setup for the selected version of MySQL Server without other products.

• **Custom**: Enables you to select any version of MySQL Server and other MySQL products.

3. Install the server instance (and products) and then begin the server configuration by first selecting one of the following levels of availability for the server instance:

• **Standalone MySQL Server / Classic MySQL Replication (default)**
  
  Configures a server instance to run without high availability.

• **InnoDB cluster**
  
  Provides two configuration options based on MySQL Group Replication to:

  • Configure multiple server instances in a sandbox InnoDB cluster on the local host (for testing only).

  • Create a new InnoDB cluster and configure one seed instance or add a new server instance to an existing InnoDB cluster.

4. Complete the configuration process by following the onscreen instructions. For more information about each individual step, see Section 1.3.3.1, “MySQL Server Configuration with MySQL Installer”.

MySQL is now installed. If you configured MySQL as a service, then Windows will automatically start MySQL server every time you restart your system. Also, this process installs the MySQL Installer application on the local host, which you can use later to upgrade or reconfigure MySQL server.

**Note**

If you installed MySQL Workbench on your system, consider using it to check your new MySQL server connection. By default, the program automatically start after installing MySQL.

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 1.4.8, “Starting MySQL as a Windows Service”.

To accommodate the `RESTART` statement, the MySQL server forks when run as a service or standalone, to enable a monitor process to supervise the server process. In this case, you will observe two `mysqld` processes. If `RESTART` capability is not required, the server can be started with the `--no-monitor` option. See `RESTART Statement`.

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`. When 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 Section 1.7, “Windows Platform Restrictions”.

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.
MySQL on Windows Considerations

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 https://dev.mysql.com/downloads/. See 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 compressed 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 1.2, “Choosing an Installation Package”.

• A MySQL Installer distribution includes MySQL Server and additional MySQL products including MySQL Workbench, MySQL for Visual Studio, and MySQL for Excel. MySQL Installer can also be used to upgrade these products in the future (see https://dev.mysql.com/doc/mysql-compat-matrix/en/).

For instructions on installing MySQL using MySQL Installer, see Section 1.3, “MySQL Installer for Windows”.

• The standard binary distribution (packaged as a compressed 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 compressed file, see Section 1.4, “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 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 Statement.

• 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 1.4.2, “Creating an Option File”.

1.1 MySQL Installation Layout on Microsoft Windows

For MySQL 8.0 on Windows, the default installation directory is `C:\Program Files\MySQL\MySQL Server 8.0` for installations performed with MySQL Installer. If you use the ZIP archive method to install MySQL, you may prefer to install in `C:\mysql`. 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 1.1 Default MySQL Installation Layout for Microsoft Windows**

<table>
<thead>
<tr>
<th>Directory</th>
<th>Contents of Directory</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>mysql server, client and utility programs</td>
<td></td>
</tr>
<tr>
<td>%PROGRAMDATA%\MySQL\MySQL Server 8.0\</td>
<td>Log files, databases</td>
<td>The Windows system variable <code>%PROGRAMDATA</code> defaults to <code>C:\ProgramData</code>.</td>
</tr>
<tr>
<td>docs</td>
<td>Release documentation</td>
<td>With MySQL Installer, use the Modify operation to select this optional folder.</td>
</tr>
<tr>
<td>include</td>
<td>Include (header) files</td>
<td></td>
</tr>
<tr>
<td>lib</td>
<td>Libraries</td>
<td></td>
</tr>
<tr>
<td>share</td>
<td>Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation</td>
<td></td>
</tr>
</tbody>
</table>

1.2 Choosing an Installation Package

For MySQL 8.0, there are multiple installation package formats to choose from when installing MySQL on Windows. The package formats described in this section are:

- MySQL Installer
- MySQL noinstall ZIP Archives
- MySQL Docker Images

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-8.0.21.0.msi` or `mysql-installer-commercial-8.0.21.0.msi`, and utilizes MSIs to automatically install MySQL server and other products. MySQL Installer will download and apply updates to itself, and for each of the installed products. It also configures the installed MySQL server (including a sandbox InnoDB cluster test setup) and MySQL Router. MySQL Installer is recommended for most users.
MySQL Installer can install and manage (add, modify, upgrade, and remove) many other MySQL products, including:

- Connectors – MySQL Connector/C++, MySQL Connector/.NET, Connector/ODBC, MySQL Connector/Python, MySQL Connector/J, MySQL Connector/Node.js
- Documentation – MySQL Manual (PDF format), samples and examples

MySQL Installer operates on all MySQL supported versions of Windows (see https://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.

For instructions on how to install MySQL using MySQL Installer, see Section 1.3, “MySQL Installer for Windows”.

MySQL noinstall ZIP Archives

These packages contain the files found in the complete MySQL Server installation package, with the exception of the GUI. This format does not include an automated installer, and must be manually installed and configured.

The noinstall ZIP archives are split into two separate compressed files. The main package is named mysql-VERSION-win64.zip. This contains the components needed to use MySQL on your system. The optional MySQL test suite, MySQL benchmark suite, and debugging binaries/information components (including PDB files) are in a separate compressed file named mysql-VERSION-win64-debug-test.zip.

If you choose to install a noinstall ZIP archive, see Section 1.4, “Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive”.

MySQL Docker Images

For information on using the MySQL Docker images provided by Oracle on Windows platform, see Deploying MySQL on Windows and Other Non-Linux Platforms with Docker.

Warning
The MySQL Docker images provided by Oracle are built specifically for Linux platforms. Other platforms are not supported, and users running the MySQL Docker images from Oracle on them are doing so at their own risk.

1.3 MySQL Installer for Windows

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

- MySQL Servers

MySQL Installer can install and manage multiple, separate MySQL server instances on the same host at the same time. For example, MySQL Installer can install, configure, and upgrade a separate instance of MySQL 5.6, MySQL 5.7, and MySQL 8.0 on the same host. MySQL Installer does not permit server upgrades between major and minor version numbers, but does permit upgrades within a release series (such as 5.7.18 to 5.7.19).
Installation Requirements

MySQL Installer cannot install both Community and Commercial (Standard Edition and Enterprise Edition) releases of MySQL server on the same host. If you require both releases on the same host, consider using the ZIP archive distribution to install one of the releases.

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

- MySQL Connectors
  MySQL Connector/NET, MySQL Connector/Python, MySQL Connector/ODBC, MySQL Connector/J, and MySQL Connector/C++.

  Note
  To install MySQL Connector/Node.js, see https://dev.mysql.com/downloads/connector/nodejs/. Connector/Node.js does not provide an .msi file for use with MySQL Installer.

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

Installation Requirements

MySQL Installer requires Microsoft .NET Framework 4.5.2 or later. If this version is not installed on the host computer, you can download it by visiting the Microsoft website.

MySQL Installer Community Release

Download software from https://dev.mysql.com/downloads/installer/ to install the Community release 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 8.0 and N is the package number, which begins at 0.

- Full or Current Bundle: Bundles all of the MySQL products for Windows (including the MySQL server). The file size is over 300 MB, and the name has the form mysql-installer-community-VERSION.N.msi where VERSION is the MySQL Server version number such as 8.0 and N is the package number, which begins at 0.

MySQL Installer Commercial Release

Download software from https://edelivery.oracle.com/ to install the Commercial (Standard Edition or Enterprise Edition) release of MySQL products for Windows. The Commercial release includes all of the current and previous GA versions in the Community release (excludes development-milestone versions) and also includes the following products:

- Workbench SE/EE
- MySQL Enterprise Backup
- MySQL Enterprise Firewall
The Commercial release integrates with your My Oracle Support (MOS) account. For knowledge-base content and patches, see My Oracle Support.

1.3.1 MySQL Installer Initial Setup

- 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 1.1 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.

### 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. By default, both GA and pre-release products are included in the download and installation with the Developer Default, Client only, and Full setup types. Select the Only install GA products option to restrict the product set to include GA products only when using these setup types.

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 Installer Initial Setup

- MySQL Connectors (for .NET / Python / ODBC / Java / C++)
- 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 `mysqldadmin`.
- **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 1.3.2, “Setting Alternative Server Paths with MySQL Installer”.

- 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
MySQL Installer Initial Setup

displays the **Check Requirements** step to help you update the host. Requirements are evaluated each time you download a new product (or version) for installation. The following figure identifies and describes the key areas of this step.

**Figure 1.2 Check Requirements**

![Figure 1.2 Check Requirements](image)

**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 by clicking **Check**.
   - **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.

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

1.3.2 Setting Alternative Server Paths with MySQL Installer

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 of MySQL Installer, 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 apply a filter on the product list shown in Available Products (see Locating Products to Install).
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 box where you can enter alternative path names. After the path names are validated, click **Next** to continue with the configuration steps.

**Figure 1.3 Change MySQL Server Path**

---

### 1.3.3 Installation Workflow with MySQL Installer

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. For assistance with product selection, see **Locating Products to Install**.

**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. If you attempt to install a product that is incompatible with the existing MySQL server version (or a version selected for upgrade), you are alerted about the possible mismatch.

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, click **Execute** to begin the download. The status of each product changes from **Downloading** to **Downloaded**.
Installation Workflow with MySQL Installer

- **Product installation.** The status of each product in the list changes from *Ready to Install*, to *Installing*, and lastly 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, MySQL Router, and samples only. The status for each item in the list should indicate *Ready to Configure*.

  Click **Next** to start the configuration wizard for all items in the list. The configuration options presented during this step are specific to the version of database or router that you selected to install.

  Click **Execute** to begin applying the configuration options or click **Back** (repeatedly) to return to each configuration page. Click **Finish** to open the MySQL Installer dashboard.

- **Installation complete.** This step finalizes the installation for products that do not require configuration. It enables you to copy the log to a clipboard and to start certain applications, such as MySQL Workbench and MySQL Shell. Click **Finish** to open the MySQL Installer dashboard.

### 1.3.3.1 MySQL Server Configuration with MySQL Installer

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

- For the MySQL 8.0 release series, a server can be configured to run as a standalone database, as a sandbox InnoDB cluster on a single host, or to create a production InnoDB cluster inside a local network (see *High Availability*).

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

  **Note**
  
  Query cache was deprecated in MySQL 5.7 and removed in MySQL 8.0 (and later).

- 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 1.3.2, “Setting Alternative Server Paths with MySQL Installer”.

- 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 `MysqlSys` 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** enables 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. Configuring MySQL server may include some or all of the steps.
High Availability

MySQL Installer enables you to install, configure, and deploy MySQL Server as a standalone instance or as a member of a highly available cluster using MySQL Group Replication. In either case, MySQL Installer restricts the installation and configuration of the server (or servers) to the local Windows host computer.

- **Standalone MySQL Server / Classic MySQL Replication (default)**

  Select this option to configure one MySQL instance to run as a standalone database server. This option is ideal if you intend to set up classic replication later and then to include this server in your custom solution. The remaining configuration steps are described in the sections that follow, starting with Type and Networking.

- **InnoDB cluster**

  Select this option to create or extend an InnoDB cluster solution that is based on MySQL Group Replication (see Introducing InnoDB Cluster). You can configure (or reconfigure) a minimum of three server instances to perform a basic setup as a test-only sandbox cluster on a single computer or to create a production cluster inside a local network.

  **InnoDB Cluster Log Verbosity Level.** This configuration step includes an adjustable log that captures information during the configuration of each server instance in the production or sandbox cluster. The values are: MINIMAL, MEDIUM (default), and DEBUG. If the cluster configuration fails, use the Reconfigure action from the MySQL Installer dashboard to restart the configuration and then set the verbosity level to DEBUG to gather additional information during your next attempt.

MySQL Installer provides the following configuration variations to deploy an InnoDB cluster:

- **Set Up a Local Server Cluster for Testing Only**

  Select Create a Sandbox InnoDB cluster for Testing to enable this option. When prompted, define the number of server sandbox instances in the cluster, set a password for the root user, and adjust the InnoDB cluster log verbosity level as needed. For a more detailed description of the configuration, see Deploying a Sandbox InnoDB Cluster with MySQL Installer. This setup requires MySQL 5.7.17 or higher.

- **Create or Join an InnoDB cluster**

  To set up a highly available InnoDB cluster using MySQL Installer, you must have a minimum of three computers on a local network. If you require a more advanced setup, use MySQL Shell to configure some or all of the server instances in the cluster. For details about how to perform a local-network cluster setup, see Setting up an InnoDB cluster with MySQL Installer. This setup requires MySQL 8.0.0 or higher.

InnoDB cluster was designed to operate with MySQL Shell, which enables you to perform advanced cluster administration, and MySQL Router to automate the connections made between client applications and server instances. Neither MySQL Shell nor MySQL Router are required to deploy a cluster on Windows using MySQL Installer.

**Deploying a Sandbox InnoDB Cluster with MySQL Installer**

A sandbox deployment includes multiple server sandbox instances that run together on the same computer. Because all server instances reside on the same computer, a sandbox cluster does not meet the requirements of a highly available solution. Instead, this deployment option simulates an environment from which you can explore the techniques associated with InnoDB cluster administration.

When you select Create a Sandbox InnoDB cluster for Testing, a follow-on step prompts you to select a cluster consisting of three, five, seven, or nine MySQL server instances. Unlike the other server setups provided by MySQL Installer, the sandbox deployment skips the usual server
Installation Workflow with MySQL Installer

configuration steps (except Authentication Method). The resulting cluster, named sandboxCluster, is available on selected ports that are configured for you.

Note
MySQL Installer deletes ports 3310 to 3390 during the configuration, if those ports were set for the sandbox InnoDB cluster manually using MySQL Shell.

Each sandbox instance is configured to run as a process (not a Windows service). You must start each instance in the sandbox cluster manually after restarting the computer.

After you create the test cluster, click the Summary tab to view the specific ports that apply to your cluster. To modify the number of server instances within the existing cluster or to adjust the logging level, use the Reconfigure quick action from the MySQL Installer dashboard.

Warning
MySQL Installer deletes all existing sandbox cluster data when the cluster is reconfigured or when the server instances within the sandbox cluster are upgraded.

MySQL Installer stores all sandbox InnoDB cluster configuration entries in the installer_config.xml file. By default, MySQL Installer creates the sandbox instances in %userprofile%\MySQL\mysql-sandboxes on the local host.

Setting up InnoDB Cluster with MySQL Installer

To create a single InnoDB cluster, select InnoDB Cluster as the High Availability option and then select Create a New InnoDB Cluster. Adjust the log verbosity level (as needed), and click Next to configure the first server instance. This setup process involves installing and running MySQL Installer on multiple computers.

Define the first server instance (or seed) by providing the following configuration information:

• InnoDB Cluster Name:
The default cluster name is myCluster. If you intend to configure multiple clusters, replace the default name with one that is meaningful within your solution. Alphanumeric characters, spaces, and underscore (_) characters are valid for this field. The limit is 40 characters.

• Cluster Admin User Name:
The default cluster administrator name is ic. You can reuse the same MySQL administrative account across multiple clusters. You will be prompted for this account name (and password) later when you configure other server instances to join the cluster. The limit is 32 characters.

• Cluster Admin Password:
Enter a password for the cluster administrator account (minimum length is four characters). MySQL Installer will evaluate the strength of the MySQL password as you type. Use the Repeat Password field to confirm the password.

• Host Address:
Select the host name or IP address of the local host from the list. When joining additional server instances to the cluster, you will be prompted to identify the seed instance by the host name or IP address.

• Server ID:
The default value is 1. This identifier is required to record the events of a server instance in the binary log. The ID of each server instance within a cluster must be unique; however, you can reuse the same number in a different cluster. The server ID you specify in this field also appears later in
Advanced Options step. If you change the value in Advanced Option, the number is changed for the InnoDB cluster Setup too.

Click Next and then complete the remaining configuration steps, which are described in the sections that follow, starting with Type and Networking. After the seed instance is added and the cluster is created, it requires more instances for full tolerance. At this point, the status is OK_NO_TOLERANCE.

To add the second and third server instances to the cluster, you must use a separate computer inside the local network for each. Some of the configuration details of the seed instance are required to complete the join operation.

After you start MySQL Installer and install the server instance on the next computer, begin the configuration by selecting InnoDB Cluster as the High Availability option and then select Add Local MySQL Server Instance to an InnoDB Cluster. Adjust the InnoDB Cluster Log Verbosity Level (as needed) and then click Next.

Define the joining server instance by providing the following configuration information:

- **Seed Instance Address:**
  Enter the host name or IP address of the computer that hosts the seed instance.

- **Seed Instance Port:**
  The default value is 3306, which is the port for classic MySQL. Use the same TCP port that you configured for the seed instance.

- **Cluster Admin User Name:**
  The default cluster administrator name is ic. If you assigned a different name when you configured the seed instance, enter the alternative cluster administrator name.

- **Cluster Admin Password:**
  Enter the password assigned to the cluster administrator account.

- **Host Address:**
  Select the host name or IP address of the local host from the list.

- **Server ID:**
  The default value is 1. This identifier is required to record the events of a server instance in the binary log. The ID of each server instance within a cluster must be unique; however, you can reuse the same number in a different cluster. The server ID you specify in this field also appears later in Advanced Options step. If you change the value in Advanced Option, the number is changed for the InnoDB cluster Setup too.

- **Test Connection**
  Use this button to verify the connection between the local server instance and the seed instance defined for the cluster. A valid connection is required to proceed.

Click Next and then complete the remaining configuration steps, which are described in the sections that follow, starting with Type and Networking.

With one seed instance and a second server instance in the cluster, the status is OK_NO_TOLERANCE. After you add the third server instance, the status is OK, which indicates that the cluster now is tolerant to the failure of one instance.

**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) to assign to your MySQL server instance.

- **Development**: A computer that hosts many other applications, and typically this is your personal workstation. This setting configures MySQL to use the least amount of memory.

- **Server**: Several other applications are expected to run on this computer, such as a web server. The Server setting configures MySQL to use a medium amount of memory.

- **Dedicated**: A computer that is dedicated to running the MySQL server. Because no other major applications run on this server, this setting configures MySQL to use the majority of available memory.

**Connectivity**

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

- **TCP/IP**: This option is selected by default. You may disable TCP/IP Networking to permit local host connections only. With the TCP/IP connection option selected, you can modify the following items:
  
  - **Port** for the classic MySQL protocol connections. The default value is **3306**.
  - **X Protocol Port** shown when configuring MySQL 8.0 server only.
  - **Open Windows Firewall port for network access**, which is selected by default for TCP/IP.

If a 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 setting the `named_pipe` system variable. The default name is **MySQL**.

- **Shared Memory**: Enable and define the memory name, similar to setting the `shared_memory` system variable. The default name is **MySQL**.

**Advanced Configuration**

Check **Show Advanced and Logging Options** to set custom logging and advanced options in later steps. The Logging Options step enables you to define 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. The Advanced Options step enables you to set the unique server ID required when binary logging is enabled in a replication topology.

- **MySQL Enterprise Firewall** (Enterprise Edition only)

The **Enable MySQL Enterprise Firewall** check box is deselected by default. Select this option to enable a security whitelist that offers protection against certain types of attacks. Additional post-installation configuration is required (see MySQL Enterprise Firewall).

**Important**

There is an issue for MySQL 8.0.19 that prevents the server from starting if MySQL Enterprise Firewall is selected during the server configuration steps. If the server startup operation fails, click **Cancel** to end the configuration process and return to the dashboard. You must uninstall the server.

The workaround is to run MySQL Installer without MySQL Enterprise Firewall selected. (That is, do not select the **Enable MySQL Enterprise Firewall** check box.) Then install MySQL Enterprise Firewall afterward using the
instructions for manual installation (see Installing or Uninstalling MySQL Enterprise Firewall).

Authentication Method

The Authentication Method step is visible only during the installation or upgrade of MySQL 8.0.4 or higher. It introduces a choice between two server-side authentication options. The MySQL user accounts that you create in the next step will use the authentication method that you select in this step.

MySQL 8.0 connectors and community drivers that use libmysqlclient 8.0 now support the mysql_native_password default authentication plugin. However, if you are unable to update your clients and applications to support this new authentication method, you can configure the MySQL server to use mysql_native_password for legacy authentication. For more information about the implications of this change, see caching_sha2_password as the Preferred Authentication Plugin.

If you are installing or upgrading to MySQL 8.0.4 or higher, select one of the following authentication methods:

- Use Strong Password Encryption for Authentication (RECOMMENDED)

MySQL 8.0 supports a new authentication based on improved, stronger SHA256-based password methods. It is recommended that all new MySQL server installations use this method going forward.

  Important

  The caching_sha2_password authentication plugin on the server requires new versions of connectors and clients, which add support for the new MySQL 8.0 default authentication.

- Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)

Using the old MySQL 5.x legacy authentication method should be considered only in the following cases:

  • Applications cannot be updated to use MySQL 8.0 connectors and drivers.
  • Recompilation of an existing application is not feasible.
  • An updated, language-specific connector or driver is not available yet.

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 (Optional)

Click Add User or Edit User to create or modify MySQL user accounts with predefined roles. Next, enter the required account credentials:

  • User Name: MySQL user names can be up to 32 characters long.
  • Host: Select localhost for local connections only or <All Hosts (%)> when remote connections to the server are required.
• **Role**: Each predefined role, such as **DB Admin**, is configured with its own set of privileges. For example, the **DB Admin** role has more privileges than the **DB Designer** role. The **Role** drop-down list contains a description of each role.

• **Password**: Password strength assessment is performed while you type the password. Passwords must be confirmed. MySQL permits a blank or empty password (considered to be insecure).

**MySQL Installer Commercial Release Only**: MySQL Enterprise Edition for Windows, a commercial product, also supports an authentication method that performs external authentication on Windows. Accounts authenticated by the Windows operating system can access the MySQL server without providing an additional password.

To create a new MySQL account that uses Windows authentication, enter the user name and then select a value for **Host** and **Role**. Click **Windows** authentication to enable the **authentication_windows** plugin. In the Windows Security Tokens area, enter a token for each Windows user (or group) who can authenticate with the MySQL user name. MySQL accounts can include security tokens for both local Windows users and Windows users that belong to a domain. Multiple security tokens are separated by the semicolon character (;) and use the following format for local and domain accounts:

• **Local account**

  Enter the simple Windows user name as the security token for each local user or group; for example, **finley;jeffrey;admin**.

• **Domain account**

  Use standard Windows syntax (**domain\domainuser**) or MySQL syntax (**domain\domainuser**) to enter Windows domain users and groups.

  For domain accounts, you may need to use the credentials of an administrator within the domain if the account running MySQL Installer lacks the permissions to query the Active Directory. If this is the case, select **Validate Active Directory users with** to activate the domain administrator credentials.

  Windows authentication permits you to test all of the security tokens each time you add or modify a token. Click **Test Security Tokens** to validate (or revalidate) each token. Invalid tokens generate a descriptive error message along with a red X icon and red token text. When all tokens resolve as valid (green text without an X icon), you can click **OK** to save the changes.

**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 account 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

### Logging 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

**Note**

The binary log is enabled by default for MySQL 5.7 and higher.

### 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.

The advanced-configuration options include:

- **Server ID**

Set the unique identifier used in a replication topology. If binary logging is enabled, you must specify a server ID. The default ID value depends on the server version. For more information, see the description of the **server_id** system variable.

**Tip**

If you specified an ID for a server instance of an InnoDB cluster, then MySQL Installer adjusts the ID (shown on this page) to match the previous identifier.

- **Table Names Case**

You can set the following options during the initial and subsequent configuration the server. For the MySQL 8.0 release series, these options apply only to the initial configuration of the server.

- **Lower Case**

  Sets the **lower_case_table_names** option value to 1 (default), in which table names are stored in lowercase on disk and comparisons are not case-sensitive.

- **Preserve Given Case**
Sets the `lower_case_table_names` option value to 2, in which table names are stored as given but compared in lowercase.

**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 (with a check mark) 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 completes successfully 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.

### 1.3.3.2 MySQL Router Configuration with MySQL Installer

MySQL Installer downloads and installs 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 8.0 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**.

**Note**

You are prompted to configure MySQL Router after the initial installation and when you reconfigure an installed router explicitly. In contrast, the upgrade operation does not require or prompt you to configure the upgraded product.

To configure MySQL Router, do the following:

1. Set up InnoDB cluster. For instructions on how to configure a sandbox InnoDB cluster on the local host using MySQL Installer, see **High Availability**.

   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**: Host name of the primary (seed) server in the InnoDB cluster (**localhost** by default).
   - **Port**: The port number of the primary (seed) server in the InnoDB cluster (**3310** by default).
   - **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 page, with the first base port number specified as 6446 and the remaining ports set by the wizard as 6447, 6448, and 6449.

Figure 1.4 MySQL Router Configuration

![MySQL Installer Product Catalog and Dashboard](image)

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

After installing a production cluster with MySQL Router, the root account only exists in the user table as `root@localhost` (local), instead of `root@%` (remote). Regardless of where the router or client are located, even if both are located on the same host as the seed server, any connection that passes through the router is viewed by server as being remote, not local. As a result, a connection made to the server using the local host (see the example that follows), does not authenticate.

```
shell> \c root@localhost:6446
```

### 1.3.4 MySQL Installer Product Catalog and Dashboard

- Product Catalog
- MySQL Installer Dashboard
- Locating Products to Install
- Upgrading MySQL Installer

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 Downloads. 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 release that you download.

MySQL Installer Dashboard

The MySQL Installer dashboard is the default view that you see when you start MySQL Installer after the initial setup finishes. If you closed MySQL Installer before the setup was finished, MySQL Installer resumes the initial setup before it displays the dashboard.

Figure 1.5 MySQL Installer Dashboard Elements

Description of MySQL Installer Dashboard Elements

1. 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 page. 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 page and populates it with all the upgrade candidates. An installed product can have more than one upgrade version and requires a current product catalog.

**Important server upgrade conditions:**

- MySQL Installer does not permit server upgrades between major release versions or minor release versions, but does permit upgrades within a release series, such as an upgrade from 5.7.18 to 5.7.19.

- Upgrades between milestone releases (or from a milestone release to a GA release) are not supported. Significant development changes take place in milestone releases and you may encounter compatibility issues or problems starting the server.

- For upgrades to MySQL 8.0.16 server and higher, a check box enables you to skip the upgrade check and process for system tables, while checking and processing data dictionary tables normally. MySQL Installer does not prompt you with the check box when the previous server upgrade was skipped or when the server was configured as a sandbox InnoDB cluster. This behavior represents a change in how MySQL Server performs an upgrade (see What the MySQL Upgrade Process Upgrades) and it alters the sequence of steps that MySQL Installer applies to the configuration process.

If you select **Skip system tables upgrade check and process. (Not recommended)**, MySQL Installer starts the upgraded server with the `--upgrade=MINIMAL` server option, which upgrades the data dictionary only. If you stop and then restart the server without the `--upgrade=MINIMAL` option, the server upgrades the system tables automatically, if needed.

The following information appears in the Log tab and log file after the upgrade configuration (with system tables skipped) is complete:

```
WARNING: The system tables upgrade was skipped after upgrading MySQL Server. The server will be started now with the --upgrade=MINIMAL option, but then each time the server is started it will attempt to upgrade the system tables, unless you modify the Windows service (command line) to add --upgrade=MINIMAL to bypass the upgrade.

FOR THE BEST RESULTS: Run mysql.exe --upgrade=FORCE on the command line to upgrade the system tables manually.
```

To choose a new product version:

a. Click **Upgrade**. 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.

**Note**

For server milestone releases in the same release series, MySQL Installer deselects the server upgrade and displays a warning to indicate that the upgrade is not supported, identifies the risks of continuing, and provides a summary of the steps to perform a logical upgrade manually. You can reselect server upgrade at your own risk. For instructions on how to perform a logical upgrade with a milestone release, see Logical Upgrade.

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 page 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. During the operation, an indicator shows the number of steps that are executed as a percentage of all steps.

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.

To remove a local MySQL server:

a. Determine whether the local data directory should be removed. If you retain the data directory, another server installation can reuse the data. This option is enabled by default (removes the data directory).

b. If the local server is a member of an InnoDB cluster, reconfigure the cluster as follows:

i. Type the administrator password for the cluster affected when the local server is removed and then click **Test Connection** to verify the credentials. MySQL Installer can perform the following actions, depending on the configuration of the existing cluster:

• If the local server is a seed instance and the number of instances in the cluster is one, dissolve the cluster when you remove the local server.

• If the local server is a seed instance and the number of instances in the cluster is greater than one, remove the instance from the cluster or dissolve the cluster when you remove the local server.

• If the local server is a slave instance within the cluster and the number of instances in the cluster is greater than two, remove the local instance from the cluster. (A single slave instance within a cluster reverts to a seed instance automatically.)

• If the local server is configured as a sandbox InnoDB cluster, remove all instances created for the sandbox server installation.

ii. When prompted, do one of the following:

• Select an action to apply to the cluster and click **Next**.

• Click **Next** without selecting an action. In most cases, MySQL Group Replication can manage the cluster when the local server becomes unavailable. A warning message reminds you that skipping the step may result in an inconsistent InnoDB cluster configuration.

c. Click **Execute** to begin uninstalling the local server. Note that all products that you selected to remove are also uninstalled at this time.

d. (Optional) Click the **Log** tab to display the current actions performed by MySQL Installer.

2. 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. You must provide credentials with root privileges to reconfigure these items. Click the **Log** tab to show the output of each configuration step performed by MySQL Installer.

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 1.3.3.1](#MySQL).
3. 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.

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

   **Tip**

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

In addition to the **About MySQL** information ( ), you can also select the following icons from the side panel:

- License icon ( ) for MySQL Installer.

   This product may include third-party software, used under license. If you are using a Commercial release of MySQL Installer, the icon opens the MySQL Installer Commercial License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a Community release of MySQL Installer, the icon opens the MySQL Installer Community License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

- Resource links icon ( ) to the latest MySQL product documentation, blogs, webinars, and more.

5. The MySQL Installer **Options** icon ( ) includes the following tabs:

   - **Product Catalog**: Manages the daily automatic catalog updates. By default, catalog updates are scheduled at a fixed hour. 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.

   - **Connectivity Settings**: Several operations performed by MySQL Installer require internet access. This option enables you to use a default value to validate the connection or to use a different URL, one selected from a list or added by you manually. With the **Manual** option selected, new URLs can be added and all URLs in the list can be moved or deleted. When the **Automatic** option is selected, MySQL Installer attempts to connect to each default URL in the list (in order) until a connection is made. If no connection can be made, it raises an error.
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-release or older version of a product, it may not be visible in the default list.

To change the default product list, click **Add** on the dashboard to open the Select Products and Features page, and then click **Edit** to open the filter dialog box (see the figure that follows). Modify the product values and then click **Filter**.

**Figure 1.6 Filter Available Products**

Reset one or more of 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).
- **Maturity**: Current Bundle (appears initially with the full package only), Pre-Release, Current GA, or Other Releases.

**Note**
The Commercial release of MySQL Installer does not display any MySQL products when you select the Pre-Release age filter. Products in development are available from the Community release of MySQL Installer only.

- **Already Downloaded** (the check box is deselected by default).
- **Architecture**: Any (default), 32-bit, or 64-bit.

Upgrading MySQL Installer

MySQL Installer remains installed on your computer, and like other software, MySQL Installer can be upgraded from the previous version. In some cases, other MySQL software may require that you upgrade MySQL Installer for compatibility. This section describes how to identify the current version of MySQL Installer and how to upgrade MySQL Installer manually.

**To locate the installed version of MySQL Installer:**

1. Start MySQL Installer from the search menu. The MySQL Installer dashboard opens.

2. Click the MySQL Installer About icon (-question mark). The version number is located above the **Back** button.
To initiate an on-demand upgrade of MySQL Installer:

1. Connect the computer with MySQL Installer installed to the internet.

2. Start MySQL Installer from the search menu. The MySQL Installer dashboard opens.

3. Click **Catalog** on the bottom of the dashboard to open the Update Catalog window.

4. Click **Execute** to begin the process. If the installed version of MySQL Installer can be upgraded, you will be prompted to start the upgrade.

5. Click **Next** to review all changes to the catalog and then click **Finish** to return to the dashboard.

6. Verify the (new) installed version of MySQL Installer (see the previous procedure).

### 1.3.5 MySQLInstallerConsole.exe 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:

```bash
C:\> cd Program Files (x86)\MySQL\MySQL Installer for Windows
C:\Program Files (x86)\MySQL\MySQL Installer for Windows> MySQLInstallerConsole.exe help
```

**MySQL Installer** is running in Community mode

------------------- Start Initialization -------------------

MySQL Installer is running in Community mode

Attempting to update manifest.

Loading product catalog

Checking for product catalog snippets

Checking for product packages in the bundle

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 commands:

**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. If pre-release products are available, both GA and pre-release products are installed when the value of the `-type` switch is `Developer`, `Client`, or `Full`. Use the `-only_ga_products` switch to restrict the product set to GA products only when using these setup types.

Switches and syntax options include:

- `-only_ga_products`: Restricts the product set to include GA products only.

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

  - **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 character (:) 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.
Adding users is not supported when a product is being reconfigured.

- **feature**: The feature block is a semicolon separated list of features, or an asterisk character (*) to select all features.

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

An example that passes in additional configuration blocks, separated by ^ to fit:

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

- **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.

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

- **modify** [product1:-removelist|+addlist] [product2:-removelist|+addlist] [...]
  Modifies or displays features of a previously installed MySQL product.
  - **-silent**: Disable confirmation prompts.

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

- **remove** [product1] [product2] [...]
  Removes one ore more products from your system.
  - **-continue**: Continue the operation even if an error occurs.
  - **-silent** : Disable confirmation prompts.

```bash
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.

```bash
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.

```bash
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

### 1.4 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 package is as follows:

1. Extract the main archive to the desired install directory

   *Optional*: also extract the debug-test archive if you plan to execute the MySQL benchmark and test suite

2. Create an option file

3. Choose a MySQL server type

4. Initialize MySQL

5. Start the MySQL server

6. Secure the default user accounts

This process is described in the sections that follow.

#### 1.4.1 Extracting the Install Archive

To install MySQL manually, do the following:

1. If you are upgrading from a previous version please refer to [Chapter 2, 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. 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 1.4.2, “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 file-compression 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.

#### 1.4.2 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 8.0` and `C:\Program Files\MySQL\MySQL Server 8.0\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 the user executing MySQL Installer is granted full permissions to this new `my.ini` file.

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 ZIP archive does not include a `data` directory. To initialize a MySQL installation by creating the data directory and populating the tables in the mysql system database, initialize MySQL using either `--initialize` or `--initialize-insecure`. For additional information, see Initializing the Data Directory.

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 8.0\data) to E:\mydata.

2. Use a --datadir option to specify the new data directory location each time you start the server.

### 1.4.3 Selecting a MySQL Server Type

The following table shows the available servers for Windows in MySQL 8.0.

<table>
<thead>
<tr>
<th>Binary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysqld</td>
<td>Optimized binary with named-pipe support</td>
</tr>
<tr>
<td>mysqld-debug</td>
<td>Like mysqld, but compiled with full debugging and automatic memory allocation checking</td>
</tr>
</tbody>
</table>

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 8.0 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 named_pipe system variable enabled. It is necessary to enable this variable 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.

### 1.4.4 Initializing the Data Directory

If you installed MySQL using the noinstall package, no data directory is included. To initialize the data directory, use the instructions at Initializing the Data Directory.

### 1.4.5 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 MySQL Installer.

The examples in these sections assume that MySQL is installed under the default location of C: \Program Files\MySQL\MySQL Server 8.0. 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 system variable enabled. Clients can connect through shared memory by using the --protocol=MEMORY option.

For information about which server binary to run, see Section 1.4.3, “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 8.0\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:

```
mysql: ready for connections
Version: '8.0.21'  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 8.0\data` by default). The error log is the file with the `.err` extension, and may be set using the `--log-error` option.

**Note**

The initial root account in the MySQL grant tables has no password. After starting the server, you should set up a password for it using the instructions in Securing the Initial MySQL Account.

### 1.4.6 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.

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 8.0\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 8.0\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 operating system 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 8.0\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 DBUG Package.

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

### 1.4.7 Customizing the PATH for MySQL Tools

**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.

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 8.0\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. The new PATH value should now be available to any new command shell you open, allowing you 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.
1.4.8 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.

The Services utility (the Windows Service Control Manager) can be found in the Windows Control Panel. 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 8.0\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 operating system users under Windows.

Install the server as a service using this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\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 8.0\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. 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 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:

```bash
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\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 operating 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

After a MySQL server instance 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 an sc start mysql_service_name or NET START mysql_service_name command. SC and NET commands are 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 8.0\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 using the Services utility, the sc stop mysql_service_name command, the NET START mysql_service_name 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 8.0\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 SC STOP mysql_service_name or NET STOP mysql_service_name. Then use SC DELETE mysql_service_name to remove it:

```
C:\> SC DELETE mysql
```

Alternatively, use the mysqld --remove option to remove the service.

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --remove
```

If mysqld is not running as a service, you can start it from the command line. For instructions, see Section 1.4.6, “Starting MySQL from the Windows Command Line”.

If you encounter difficulties during installation, see Section 1.5, “Troubleshooting a Microsoft Windows MySQL Server Installation”.

For more information about stopping or removing a Windows service, see Starting Multiple MySQL Instances as Windows Services.

1.4.9 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 8.0\bin\mysqlshow"
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqlshow" -u root mysql
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin" version status proc
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\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` system variable enabled 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 Using Command Options.

For more information about `mysqlshow`, see `mysqlshow — Display Database, Table, and Column Information`.

### 1.5 Troubleshooting a Microsoft Windows MySQL ServerInstallation

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 8.0\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 `SC START mysql_service_name` or `NET START mysql_service_name` 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 1.4.8, “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 8.0\data` and `C:\Program Files\MySQL\MySQL Server 8.0\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 8.0`, 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:

```plaintext
[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:

```plaintext
[mysqld]
# set basedir to your installation path
basedir=C:\Program Files\MySQL\MySQL Server 8.0
# 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 1.4.2, “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](http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/delsrv-o.asp) and use the `delsrv mysql` syntax.

### 1.6 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.

If necessary, initialize the data directory and create the MySQL grant tables. Windows installation operations performed by MySQL Installer initialize the data directory automatically. For installation from a ZIP Archive package, initialize the data directory as described at Initializing the Data Directory.

Regarding passwords, if you installed MySQL using the MySQL Installer, you may have already assigned a password to the initial `root` account. (See Section 1.3, “MySQL Installer for Windows”.) Otherwise, use the password-assignment procedure given in Securing the Initial MySQL Account.
Before assigning a password, 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 1.4.5, “Starting the Server for the First Time”). You can also set up a MySQL service that runs automatically when Windows starts (see Section 1.4.8, “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 1.3, “MySQL Installer for Windows”), the default installation directory is

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 8.0"
```

A common installation location for installation from a ZIP archive 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 1.4.7, “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 |
| sys                |
+--------------------+
```

The list of installed databases may vary, but always includes at least 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 install MySQL using MySQL Installer, the root user is 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 |
| sys                |
+--------------------+
```

If you specify a database name, `mysqlshow` displays a list of the tables within the database:

```
C:\> bin\mysqlshow mysql
Database: mysql
+---------------+
| Tables        |
+---------------+
```
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 | caching_sha2_password |
+-------+-----------+-----------------------+
```

For more information about `mysql` and `mysqlshow`, see `mysql — The MySQL Command-Line Client`, and `mysqlshow — Display Database, Table, and Column Information`.

### 1.7 Windows Platform Restrictions

The following restrictions apply to use of MySQL on the Windows platform:

- **Process memory**

  On Windows 32-bit platforms, it is not possible by default to use more than 2GB of RAM within a single process, including MySQL. This is because the physical address limit on Windows 32-bit is 4GB and the default setting within Windows is to split the virtual address space between kernel (2GB) and user/applications (2GB).

  Some versions of Windows have a boot time setting to enable larger applications by reducing the kernel application. Alternatively, to use more than 2GB, use a 64-bit version of Windows.

- **File system aliases**

  When using MyISAM tables, you cannot use aliases within Windows link to the data files on another volume and then link back to the main MySQL `datadir` location.
Windows Platform Restrictions

This facility is often used to move the data and index files to a RAID or other fast solution.

- **Limited number of ports**

  Windows systems have about 4,000 ports available for client connections, and after a connection on a port closes, it takes two to four minutes before the port can be reused. In situations where clients connect to and disconnect from the server at a high rate, it is possible for all available ports to be used up before closed ports become available again. If this happens, the MySQL server appears to be unresponsive even though it is running. Ports may be used by other applications running on the machine as well, in which case the number of ports available to MySQL is lower.

  For more information about this problem, see [http://support.microsoft.com/default.aspx?scid=kb;en-us;196271](http://support.microsoft.com/default.aspx?scid=kb;en-us;196271).

- **DATA DIRECTORY and INDEX DIRECTORY**

  The **DATA DIRECTORY** clause of the **CREATE TABLE** statement is supported on Windows for InnoDB tables only, as described in Creating Tables Externally. For MyISAM and other storage engines, the **DATA DIRECTORY** and **INDEX DIRECTORY** clauses for CREATE TABLE are ignored on Windows and any other platforms with a nonfunctional `realpath()` call.

- **DROP DATABASE**

  You cannot drop a database that is in use by another session.

- **Case-insensitive names**

  File names are not case-sensitive on Windows, so MySQL database and table names are also not case-sensitive on Windows. The only restriction is that database and table names must be specified using the same case throughout a given statement. See Identifier Case Sensitivity.

- **Directory and file names**

  On Windows, MySQL Server supports only directory and file names that are compatible with the current ANSI code pages. For example, the following Japanese directory name will not work in the Western locale (code page 1252):

  ```
  datadir="C:/私たちのプロジェクトのデータ"
  ```

  The same limitation applies to directory and file names referred to in SQL statements, such as the data file path name in **LOAD DATA**.

- **The \ path name separator character**

  Path name components in Windows are separated by the `\` character, which is also the escape character in MySQL. If you are using **LOAD DATA** or **SELECT ... INTO OUTFILE**, use Unix-style file names with `/` characters:

  ```
  mysql> LOAD DATA INFILE 'C:/tmp/skr.txt' INTO TABLE skr;
  mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
  ```

  Alternatively, you must double the `\` character:

  ```
  mysql> LOAD DATA INFILE 'C:\tmp\skr.txt' INTO TABLE skr;
  mysql> SELECT * INTO OUTFILE 'C:\tmp\skr.txt' FROM skr;
  ```

- **Problems with pipes**

  Pipes do not work reliably from the Windows command-line prompt. If the pipe includes the character `^Z / CHAR(24)`, Windows thinks that it has encountered end-of-file and aborts the program.

  This is mainly a problem when you try to apply a binary log as follows:
If you have a problem applying the log and suspect that it is because of a `^Z / CHAR(24)` character, you can use the following workaround:

```
C:\> mysqlbinlog binary_log_file | mysql --user=root
C:\> mysqlbinlog binary_log_file --result-file=/tmp/bin.sql
C:\> mysql --user=root --execute "source /tmp/bin.sql"
```

The latter command also can be used to reliably read any SQL file that may contain binary data.
Chapter 2 Upgrading MySQL on Windows

There are two approaches for upgrading MySQL on Windows:

- Using MySQL Installer
- Using the Windows ZIP archive distribution

The approach you select depends on how the existing installation was performed. Before proceeding, review Upgrading MySQL for additional information on upgrading MySQL that is not specific to Windows.

Note

Whichever approach you choose, always back up your current MySQL installation before performing an upgrade. See Database Backup Methods.

Upgrades between non-GA releases (or from a non-GA release to a GA release) are not supported. Significant development changes take place in non-GA releases and you may encounter compatibility issues or problems starting the server.

Note

MySQL Installer does not support upgrades between Community releases and Commercial releases. If you require this type of upgrade, perform it using the ZIP archive approach.

Upgrading MySQL with MySQL Installer

Performing an upgrade with MySQL Installer is the best approach when the current server installation was performed with it and the upgrade is within the current release series. MySQL Installer does not support upgrades between release series, such as from 5.7 to 8.0, and it does not provide an upgrade indicator to prompt you to upgrade. For instructions on upgrading between release series, see Upgrading MySQL Using the Windows ZIP Distribution.

To perform an upgrade using MySQL Installer:

1. Start MySQL Installer.
2. From the dashboard, click Catalog to download the latest changes to the catalog. The installed server can be upgraded only if the dashboard displays an arrow next to the version number of the server.
3. Click Upgrade. All products that have a newer version now appear in a list.

Note

MySQL Installer deselects the server upgrade option for milestone releases (Pre-Release) in the same release series. In addition, it displays a warning to indicate that the upgrade is not supported, identifies the risks of continuing, and provides a summary of the steps to perform an upgrade manually. You can reselect server upgrade and proceed at your own risk.

4. Deselect all but the MySQL server product, unless you intend to upgrade other products at this time, and click Next.
5. Click Execute to start the download. When the download finishes, click Next to begin the upgrade operation.

Upgrades to MySQL 8.0.16 and higher may show an option to skip the upgrade check and process for system tables. For more information about this option, see Important server upgrade conditions.
6. Configure the server.

**Upgrading MySQL Using the Windows ZIP Distribution**

To perform an upgrade using the Windows ZIP archive distribution:

1. Download the latest Windows ZIP Archive distribution of MySQL from https://dev.mysql.com/downloads/.

2. If the server is running, stop it. If the server is installed as a service, stop the service with the following command from the command prompt:

   ```
   C:\> SC STOP mysqld_service_name
   ```

   Alternatively, use `NET STOP mysqld_service_name`.

   If you are not running the MySQL server as a service, use `mysqladmin` to stop it. For example, before upgrading from MySQL 5.7 to 8.0, use `mysqladmin` from MySQL 5.7 as follows:

   ```
   C:\> "C:\Program Files\MySQL\MySQL Server 5.7\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.

3. Extract the ZIP archive. You may either overwrite your existing MySQL installation (usually located at `C:\mysql`), or install it into a different directory, such as `C:\mysql8`. Overwriting the existing installation is recommended.

4. Restart the server. For example, use the `SC START mysqld_service_name` or `NET START mysqld_service_name` command if you run MySQL as a service, or invoke `mysqld` directly otherwise.

5. Prior to MySQL 8.0.16, run `mysql_upgrade` as Administrator 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`. As of MySQL 8.0.16, this step is not required, as the server performs all tasks previously handled by `mysql_upgrade`.

6. If you encounter errors, see Section 1.5, “Troubleshooting a Microsoft Windows MySQL Server Installation”.
Chapter 3 Connection to MySQL Server Failing on Windows

When you’re running a MySQL server on Windows with many TCP/IP connections to it, and you’re experiencing that quite often your clients get a **Can't connect to MySQL server** error, the reason might be that Windows does not allow for enough ephemeral (short-lived) ports to serve those connections.

The purpose of **TIME_WAIT** is to keep a connection accepting packets even after the connection has been closed. This is because Internet routing can cause a packet to take a slow route to its destination and it may arrive after both sides have agreed to close. If the port is in use for a new connection, that packet from the old connection could break the protocol or compromise personal information from the original connection. The **TIME_WAIT** delay prevents this by ensuring that the port cannot be reused until after some time has been permitted for those delayed packets to arrive.

It is safe to reduce **TIME_WAIT** greatly on LAN connections because there is little chance of packets arriving at very long delays, as they could through the Internet with its comparatively large distances and latencies.

Windows permits ephemeral (short-lived) TCP ports to the user. After any port is closed it will remain in a **TIME_WAIT** status for 120 seconds. The port will not be available again until this time expires. The default range of port numbers depends on the version of Windows, with a more limited number of ports in older versions:

- Windows through Server 2003: Ports in range 1025–5000

With a small stack of available TCP ports (5000) and a high number of TCP ports being open and closed over a short period of time along with the **TIME_WAIT** status you have a good chance for running out of ports. There are two ways to address this problem:

- Reduce the number of TCP ports consumed quickly by investigating connection pooling or persistent connections where possible
- Tune some settings in the Windows registry (see below)

**Important**

The following procedure involves modifying the Windows registry. Before you modify the registry, make sure to back it up and make sure that you understand how to restore it if a problem occurs. For information about how to back up, restore, and edit the registry, view the following article in the Microsoft Knowledge Base: [http://support.microsoft.com/kb/256986/EN-US/](http://support.microsoft.com/kb/256986/EN-US/).

1. Start Registry Editor (Regedt32.exe).
2. Locate the following key in the registry:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
   ```

3. On the **Edit** menu, click **Add Value**, and then add the following registry value:

   ```
   Value Name: MaxUserPort
   Data Type: REG_DWORD
   Value: 65534
   ```

   This sets the number of ephemeral ports available to any user. The valid range is between 5000 and 65534 (decimal). The default value is 0x1388 (5000 decimal).

4. On the **Edit** menu, click **Add Value**, and then add the following registry value:

   ```
   Value Name: TcpTimedWaitDelay
   ```
Data Type: REG_DWORD
Value: 30

This sets the number of seconds to hold a TCP port connection in **TIME_WAIT** state before closing. The valid range is between 30 and 300 decimal, although you may wish to check with Microsoft for the latest permitted values. The default value is 0x78 (120 decimal).

5. Quit Registry Editor.

6. Reboot the machine.

Note: Undoing the above should be as simple as deleting the registry entries you’ve created.
Chapter 4 Resetting the Root Password: Windows Systems

On Windows, use the following procedure to reset the password for the MySQL 'root'@'localhost' account. To change the password for a root account with a different host name part, modify the instructions to use that host name.

1. Log on to your system as Administrator.

2. Stop the MySQL server if it is running. For a server that is running as a Windows service, go to the Services manager: From the Start menu, select Control Panel, then Administrative Tools, then Services. Find the MySQL service in the list and stop it.

   If your server is not running as a service, you may need to use the Task Manager to force it to stop.

3. Create a text file containing the password-assignment statement on a single line. Replace the password with the password that you want to use.

   ```sql
   ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass';
   ```

4. Save the file. This example assumes that you name the file `C:\mysql-init.txt`.

5. Open a console window to get to the command prompt: From the Start menu, select Run, then enter `cmd` as the command to be run.

6. Start the MySQL server with the `init_file` system variable set to name the file (notice that the backslash in the option value is doubled):

   ```
   C:\> cd "C:\Program Files\MySQL\MySQL Server 8.0\bin"
   C:\> mysqld --init-file=C:\mysql-init.txt
   ```

   If you installed MySQL to a different location, adjust the `cd` command accordingly.

   The server executes the contents of the file named by the `init_file` system variable at startup, changing the 'root'@'localhost' account password.

   To have server output to appear in the console window rather than in a log file, add the `--console` option to the `mysqld` command.

   If you installed MySQL using the MySQL Installation Wizard, you may need to specify a `--defaults-file` option. For example:

   ```
   C:\> mysqld
   --defaults-file="C:\ProgramData\MySQL\MySQL Server 8.0\my.ini"
   --init-file=C:\mysql-init.txt
   ```

   The appropriate `--defaults-file` setting can be found using the Services Manager: From the Start menu, select Control Panel, then Administrative Tools, then Services. Find the MySQL service in the list, right-click it, and choose the Properties option. The Path to executable field contains the `--defaults-file` setting.

7. After the server has started successfully, delete `C:\mysql-init.txt`.

   You should now be able to connect to the MySQL server as root using the new password. Stop the MySQL server and restart it normally. If you run the server as a service, start it from the Windows Services window. If you start the server manually, use whatever command you normally use.
Chapter 5 MySQL Notifier

Table of Contents

5.1 MySQL Notifier Overview ................................................................. 51
5.1.1 MySQL Notifier Tasks ................................................................. 51
5.1.2 MySQL Notifier Options ............................................................. 52
5.1.3 Managing Monitored Items ......................................................... 53
5.1.4 Troubleshooting ........................................................................... 59
5.2 Setting Up Remote Monitoring in MySQL Notifier ............................. 59

5.1 MySQL Notifier Overview

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

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

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

For notes detailing the changes in each release of MySQL Notifier, see the MySQL Notifier Release Notes.

Visit the MySQL Notifier forum for additional MySQL Notifier help and support.

Features Include:

• Start, stop, and restart instances of the MySQL server.

• Automatically detects (and adds) new MySQL server services. These are listed under Manage Monitored Items, and may also be configured.

• The Tray icon changes, depending on the status. It is a right-pointing green triangle if all monitored MySQL server instances are running or a red square if at least one service is stopped. The Update MySQL Notifier tray icon based on service status option, which dictates this behavior, is enabled by default for each service.

• Links to other applications like MySQL Workbench, MySQL Installer, and the MySQL Utilities. For example, choosing Manage Instance will load the MySQL Workbench Server Administration window for that particular instance.

• If MySQL Workbench is also installed, then the Manage Instance and SQL Editor options are available for local (but not remote) MySQL instances.

• Monitors both local and remote MySQL instances.

5.1.1 MySQL Notifier Tasks

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

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

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

Figure 5.1 MySQL Notifier Service Instance Menu

The Actions menu includes the following items:

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

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

5.1.2 MySQL Notifier Options

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

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

General Options. This group includes:

- Use colorful status icons: Enables a colorful style of icons for the tray of MySQL Notifier. Selected by default.
- Run at Windows Startup: Allows the application to be loaded when Microsoft Windows starts. Deselected by default.
- Automatically Check For Updates Every # Days: Checks for a new version of MySQL Notifier, and runs this check every # days (1 to 365). This configurable option is enabled by default.
• **Automatically add new services whose name contains**: The text used to filter services and add them automatically to the monitored list of the local computer running MySQL Notifier and on remote computers already monitoring Windows services. Selected by default for names containing `mysql`.

• **Ping monitored MySQL Server instances every # seconds**: The interval (in seconds) to ping monitored MySQL Server instances for status changes. Longer intervals might be necessary if the list of monitored remote instances is large. 30 seconds by default.

**Notification Options.** This group includes:

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

• **Notify me when a service changes status**: Displays a balloon notification from the taskbar when a monitored service changes its status. Selected by default.

**MySQL Server Connections Options.** This group includes:

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

### 5.1.3 Managing Monitored Items

Manage Monitored Items in the Actions menu enables you to add, configure, and delete the services and MySQL instances you intend to monitor. The Manage Items window has two tabs: Services and Instances. This section describes the following aspects of Managed Items window:

• Services Tab

• Instances Tab

• MySQL Server Connections

**Services Tab**

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

• **Notify me when status changes**

• **Update MySQL Notifier tray icon based on service status**

The next figure shows the Monitored Items windows with the Services tab open. This tab lists the service name, the computer where the service is hosted, and the current status of each service monitored by MySQL Notifier.
Managing Monitored Items

**Figure 5.2 MySQL Notifier: Manage Monitored Services**

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

To begin monitoring a service, click **Add** to open the Add Service window. Select a computer from the drop-down list, locate and select the service to be monitored from the list, and then click **OK**. Use the **Filter** field to reduce the set of services in the list or select **Only show services that match auto-add filter** to reuse the general-options filter from the **Options** menu.

A variety of Windows services (including MySQL) may be selected as the following figure shows. In addition to the service name, the list shows the current status of each Windows services for the selected computer.
When MySQL is configured as a MySQL instance, MySQL Notifier adds the instance to the **Instances** tab automatically. With the **Instances** tab open, you can select the following options that apply to each instance being monitored:

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

The next figure shows the **Instances** tab open and both options selected. Monitoring the instance status is set to every two minutes in this example. This tab shows the instance name, the database driver, and the current status of the instance.
To stop monitoring an instance, select it from the list of monitored MySQL instances and click **Delete**.

To begin monitoring an instance, click **Add** to open the Monitor MySQL Server Instance window. Use the **Filter** field to reduce the set of instances in the list or select **Show MySQL instances already being monitored** to show monitored items only.

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

New server connections are configured in the MySQL Server Connection window. Connection names must be unique. An alert icon (!) indicates that an option value is required. The figure that follows shows the MySQL Server Connection window with the **Parameters** tab selected.

**Figure 5.6 MySQL Notifier: MySQL Server Connection**
For each connection, provide the connection details, click **Test Connection** to confirm the MySQL connection is valid, and click **OK** to save the new connection. The type of connection you create can vary depending on the configuration of the server, the client host computer, and the level of security you want. MySQL Notifier supports the following connection types:

**Basic connections.** A basic connection is either unencrypted or encrypted (in MySQL 8.0, SSL is enabled by default) and the connection is made using standard TCP/IP, which is the default connection method in MySQL Notifier to connect to the MySQL RDBMS. Basic connections are easy to configure, particularly if MySQL Notifier and the MySQL server are on the same host computer or operate within the same local area network. To configure a basic connection, set the **Connection Method** option to **TCP/IP (standard)** and use the **Parameters** tab to configure the connection.

**SSL connections.** Both the MySQL server and the client must be configured to enable SSL encryption (see Using Encrypted Connections). To configure this type of connection, set the **Connection Method** option to **TCP/IP (standard)** and use the **Parameters** tab to configure the basic connection. Next, select the **SSL** tab to identify the appropriate files. MySQL Server uses the PEM format for certificates and private keys. In addition to providing the paths to certificate files, you can specify the SSL mode to use for your connection. The following table describes each **Use SSL** option value and indicates which files are required.

<table>
<thead>
<tr>
<th>Option value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Do not use SSL. No SSL files are required.</td>
</tr>
<tr>
<td>If Available</td>
<td>Use SSL if the server supports it, but allow connection in all cases. No SSL files are required; however, providing the SSL CA file is the best practice for connections made to MySQL 8.0 servers.</td>
</tr>
<tr>
<td>Require</td>
<td>Always use SSL and deny a connection if the server does not support SSL. Do not perform server certificate validation. No SSL files are required.</td>
</tr>
<tr>
<td>Require and Verify CA</td>
<td>Always use SSL. Validate the certificate authorities (CA), but tolerate a name mismatch. Requires the SSL CA file.</td>
</tr>
<tr>
<td>Require and Verify Identity</td>
<td>Always use SSL and fail if the host name is not correct. Requires valid SSL CA, SSL Cert, and SSL Key files.</td>
</tr>
</tbody>
</table>

**SSH connections.** SSH tunnels permit you to connect to a MySQL database from behind a firewall when the MySQL server port is blocked. To configure this type of connection, set the **Connection Method** option to **Standard TCP/IP over SSH** and use the **Parameters** tab to configure the connection.

Additional considerations:

- The SSH server communicates with a MySQL server instance in an unencrypted or encrypted mode, based on the value selected for the SSL mode (see Table 5.1, “Use SSL Option Values”). Select the **SSL** tab to specify a value for the **Use SSL** option.

- The account you use to run MySQL Notifier must be authorized on the SSH server in advance. To authenticate your account (specified by **SSH User Name**), you can use one of the following strategies:
  - **SSH User Name and SSH Password** only
  - **SSH User Name and SSH Key File** only

  Your SSH server may require that you also provide the **SSH Passphrase** property when using a key file. An invalid pass phrase generates an exception.

- **SSH User Name and SSH Key File (SSH Passphrase) and SSH Password**
Troubleshooting

The combination of user name + key file (passphrase) + password can perform fallback authentication when the key file and pass phrase are valid, but an error occurs on the server. Specifically, the first attempt to connect uses the key file, and if it fails, the next attempt to connect uses the password instead. If the SSH key file is null or empty, but the SSH password is provided, MySQL Notifier attempts to connect using the SSH password only.

5.1.4 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 MySQL VERSION failed the most recent status change request with the message "The service mysql VERSION was not found in the Windows Services". Explanation: this is a case-sensitivity issue, in that the service name is MySQL VERSION compared to having mysql VERSION in the configuration file.

Solution: either update your MySQL Notifier configuration file with the correct information, or stop MySQL Notifier and delete this configuration file. The MySQL Notifier configuration file is located at %APPDATA%@Oracle\MySQL Notifier\settings.config where %APPDATA@ is a variable and depends on your system. A typical location is "C:\Users\YourUsername\AppData\Roaming\Oracle\MySQL Notifier\settings.config" where YourUsername is your system user name. In this file, and within the ServerList section, change the ServerName values from lowercase to the actual service names. For example, change mysql VERSION to MySQL VERSION, 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 window 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 5.2, “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.

5.2 Setting Up Remote Monitoring in MySQL Notifier

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

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

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

\textbf{Figure 5.7 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 \textit{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).

\textbf{Computer running MySQL Notifier (Computer A)}

1. Enable remote administration by either editing the \textbf{Group Policy Editor}, or using \textbf{NETSH}:
   
   Using the \textbf{Group Policy Editor}:
   
   a. Click \textbf{Start}, click \textbf{Run}, type \texttt{GPEDIT.MSC}, and then click \textbf{OK}.
   
   b. Under the \textbf{Local Computer Policy} heading, expand \textbf{Computer Configuration}.
   
   c. Expand \textbf{Administrative Templates}, then \textbf{Network}, \textbf{Network Connections}, and then \textbf{Windows Firewall}.
   
   d. If the computer is in the domain, then double-click \textbf{Domain Profile}; otherwise, double-click \textbf{Standard Profile}.
   
   e. Double-click \textbf{Windows Firewall: Allow inbound remote administration exception} to open a configuration window.
   
   f. Check the \textbf{Allowed} option button and then click \textbf{OK}.

   Using the \textbf{NETSH} command:

   \textbf{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 \textbf{Run as Administrator}).

   b. Execute the following command:

   \texttt{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 \textbf{Run as Administrator}).
b. Execute the following command:

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

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

Using the Windows Firewall configuration:

a. In the Control Panel, double-click **Windows Firewall**.

b. In the Windows Firewall window, click **Allow a program or feature through Windows Firewall**.

c. In the Allowed Programs window, click **Change Settings** and do one of the following:

   - If **MySqlNotifier.exe** is in the Allowed programs and features list, make sure it is checked for the type of networks the computer connects to (Private, Public or both).
   
   - If **MySqlNotifier.exe** is not in the list, click **Allow another program**.

   i. In the **Add a Program** window, select the **MySqlNotifier.exe** if it exists in the Programs list, otherwise click **Browse** and go to the directory where **MySqlNotifier.exe** was installed to select it, then click **Add**.

   ii. Make sure **MySqlNotifier.exe** is checked for the type of networks the computer connects to (Private, Public or both).

Using the **NETSH** command:

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

b. Execute the following command, where you change "[YOUR_INSTALL_DIRECTORY]":

```bash
NETSH advfirewall firewall add rule name=MySqlNotifier program=[YOUR_INSTALL_DIRECTORY]\MySqlNotifier.exe action=allow dir=in
```

4. If Computer B is either a member of **WORKGROUP** or is in a different domain that is untrusted by Computer A, then the callback connection (Connection 2) is created as an Anonymous connection. To grant Anonymous connections DCOM Remote Access permissions:

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

b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.

c. In the My Computer Properties dialog box, click the **COM Security** tab.

d. Under Access Permissions, click **Edit Limits**.

e. In the Access Permission dialog box, select **ANONYMOUS LOGON name** in the Group or user names box. In the Allow column under Permissions for User, select **Remote Access**, and then click **OK**.

**Monitored Remote Computer (Computer B)**

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

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

Grant DCOM remote access permissions:
   a. Click **Start**, click **Run**, type **DCOMCNFG**, and then click **OK**.
   b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.
   c. In the My Computer Properties dialog box, click the **COM Security** tab.
   d. Under Access Permissions, click **Edit Limits**.
   e. In the Access Permission dialog box, select ANONYMOUS LOGON name in the Group or user names box. In the Allow column under Permissions for User, select **Remote Access**, and then click **OK**.

2. Allowing non-administrator users access to a specific WMI namespace:
   a. In the Control Panel, double-click **Administrative Tools**.
   b. In the Administrative Tools window, double-click **Computer Management**.
   c. In the Computer Management window, expand the **Services and Applications** tree.
   d. Right-click the WMI Control icon and select **Properties**.
   e. In the WMI Control Properties window, click the **Security** tab.
   f. In the Security tab, select the namespace and click **Security**. Root/CIMV2 is a commonly used namespace.
   g. Locate the appropriate account and check **Remote Enable** in the Permissions list.
3. Allow for remote administration by either editing the Group Policy Editor or using NETSH:

Using the Group Policy Editor:

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

Using the NETSH command:

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

```plaintext
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
Common Errors

- Access to the remote WMI namespace was not configured properly (see Computer B, the Allowing non-administrator users access to a specific WMI namespace step).

- 0x800706BA

- The DCOM port is not open on the client computers (Computer A) firewall. See the Open the DCOM port TCP 135 step for Computer A.

- The remote computer (Computer B) is inaccessible because its network location is set to Public. Make sure you can access it through the Windows Explorer.
Chapter 6 MySQL for Excel

Table of Contents

6.1 Introduction ................................................................. 65
6.2 Installation .................................................................. 66
6.3 Configuration ................................................................. 67
  6.3.1 MySQL Connections in Excel ...................................... 67
  6.3.2 New Schema Creation in Excel ................................. 70
  6.3.3 Global Options ......................................................... 71
6.4 What is New in MySQL for Excel .................................... 77
  6.4.1 What is New in MySQL for Excel 1.3 ......................... 77
  6.4.2 What is New in MySQL for Excel 1.2 ......................... 79
6.5 Edit MySQL Data in Excel ................................................. 80
  6.5.1 Edit Session Overview .............................................. 80
  6.5.2 Multiple Edit Sessions ............................................. 82
6.6 Import MySQL Data into Excel ........................................ 82
  6.6.1 Importing a Procedure ............................................. 84
  6.6.2 Import — Advanced Options .................................... 85
  6.6.3 Adding Summary Fields ........................................... 87
  6.6.4 Creating PivotTables ............................................... 88
6.7 Append Excel Data into MySQL ......................................... 97
  6.7.1 Column Mappings ................................................... 97
  6.7.2 Append — Advanced Options ................................... 99
6.8 Export Excel Data into MySQL .......................................... 102
  6.8.1 Column Options ..................................................... 103
  6.8.2 Export — Advanced Options ................................... 103
  6.8.3 Additional Notes .................................................... 106
6.9 MySQL for Excel Frequently Asked Questions ...................... 106

6.1 Introduction

MySQL for Excel is an add-in that enables you to browse MySQL schemas, tables, views, and procedures from within Microsoft Excel. With MySQL for Excel, you can perform the following operations:

- Import MySQL data into Excel.
- Export Excel data to MySQL as a new table or append data to an existing table.
- Modify MySQL data directly from within Excel.

**Known limitation for binary data.** Import data and export data operations ignore columns containing binary data (Varbinary or Blob). This limitation is due to the following factors:

- Binary data within Excel cells have undefined characteristics by design.
- Excel cells use text and numbers instead of binary data in a native form. Likewise, dates are stored as numbers internally in Excel.
- Excel cells restrict the size of data that they can store, which a binary data field is likely to exceed.

External Resources That Supplement the Documentation

- **Release Notes**: for notes detailing the changes in each release of MySQL for Excel, see MySQL for Excel Release Notes.
- **Support Forum**: visit the MySQL for Excel forum for additional MySQL for Excel help and support.
• **Video Introduction**: for a short introductory to MySQL for Excel video, see Video: MySQL for Excel Introduction on the official MySQL YouTube channel.

• **Developer Blog**: developers of MySQL for Excel post related blog posts on the official MySQL on Windows Blog.

### 6.2 Installation

MySQL for Excel is a 32-bit add-in for Microsoft Excel, which you can install with MySQL Installer for Windows. MySQL for Excel does not run on OS X or macOS.

MySQL for Excel can interact with MySQL Workbench to simplify the management of MySQL connections when both MySQL client tools are installed. MySQL Workbench is not required to perform database operations from MySQL for Excel.

### MySQL for Excel Requirements

- .NET Framework 4.5.2 (Client or Full Profile).
- Microsoft Office Excel 2007 or later (for Microsoft Windows).
- Visual Studio 2010 Tools for Office Runtime. MySQL Installer may install this for you.

**Note**

This requirement is different than Office Developer Tools for Visual Studio, which is not a substitute.

- An available MySQL server connection.

### MySQL for Excel Download and Install

There are two download options available for installing MySQL for Excel:

- **Standalone**: Download and execute the MySQL for Excel standalone MSI file. With this option, you must ensure that all MySQL for Excel requirements are met before executing the MSI file.

- **MySQL Installer**: Download and execute the MySQL Installer MSI file. This option is recommended because, in addition to managing all MySQL client applications for Windows, MySQL Installer evaluates the installation requirements on your computer for you.

  - When MySQL Installer is installed for the first time: Use the Developer, Client Only, or Full setup types to install MySQL for Excel together with other MySQL applications. Use the Custom setup type to install MySQL for Excel alone.

  - When MySQL Installer is already installed: Use the Add operation to install MySQL for Excel.

For additional installation and usage instructions, see Section 1.3, “MySQL Installer for Windows”.

### Open MySQL for Excel

MySQL for Excel is loaded and executed by selecting the Data menu tab in Excel, and then clicking the MySQL for Excel button from the ribbon. This button opens the MySQL for Excel task pane from which you can select options and operations. Initially, MySQL for Excel displays connection information, as shown in the following figure.

**Note**

If the items in the task pane overlap, you may need to restart Microsoft Windows after installing MySQL for Excel.
6.3 Configuration

This section describes how to configure new connections to local or remote MySQL servers from within Microsoft Excel and how to manage your existing connections from Excel or with the help of MySQL Workbench (if installed). It further describes how to create and configure a new schema (or database) from Excel.

MySQL for Excel global options that apply to all connections, SQL queries, spatial data, imported tables, and edit sessions are explained in this section.

6.3.1 MySQL Connections in Excel

MySQL for Excel provides several options to create and manage MySQL connections. You must open a connection to a MySQL server before you can configure global options, add a new schema, or perform operations that move data between Excel worksheets and MySQL tables. This section describes how to add or modify connections to MySQL.

- Adding New MySQL Connections
- Editing MySQL Connections

As the following figure shows, the MySQL for Excel task pane displays connection actions by default when it opens.
MySQL Connections in Excel

Figure 6.2 MySQL for Excel: MySQL Connections

Description of MySQL for Excel Connection Elements

1. **Open a MySQL Connection** lists the existing local and remote MySQL server connections. When you install MySQL for Excel, it creates a local connection automatically for each local MySQL server instance that you have configured on the system. After the initial installation, you can add new local or remote connections to the list.

   Double-click a connection from the list to open it. The MySQL server associated with the connection must be started on the local or remote host before you can open the connection. Select a schema to view and then click **Next**. Click **Back** to return to the list of schemas.

   Right-click an existing connection in the list to delete, edit, or refresh it.

2. **New Connection** opens a connection dialog in which you can define and test a new connection. For instructions, see Adding MySQL Connections.

3. **Manage Connections** launches MySQL Workbench if it is installed on the local host.

   MySQL for Excel shares its MySQL connections with MySQL Workbench, although it is optional to have MySQL Workbench installed. Creating and editing MySQL connections in either application will edit the MySQL connection information for both applications.

Adding New MySQL Connections

You can use MySQL for Excel or MySQL Workbench to add new MySQL connections. Adding new connections is not permitted when MySQL Workbench is open.

To add a new connection, click **New Connection** in the MySQL for Excel task pane to open the MySQL Server Connection dialog. Connection names must be unique. An alert icon (!) indicates that an option value is required. The figure that follows shows the connection dialog with the **Parameters** tab selected.
For each connection, provide the connection details, click **Test Connection** to confirm the MySQL connection is valid, and click **OK** to save the new connection. The type of connection you create can vary depending on the configuration of the server, the client host computer, and the level of security you want. MySQL for Excel supports the following connection types:

**Basic connections.** A basic connection is either unencrypted or encrypted (in MySQL 8.0, SSL is enabled by default) and the connection is made using standard TCP/IP, which is the default connection method in MySQL for Excel to connect to the MySQL RDBMS. Basic connections are easy to configure, particularly if MySQL for Excel and the MySQL server are on the same host computer or operate within the same local area network. To configure a basic connection, set the **Connection Method** option to **TCP/IP (standard)** and use the **Parameters** tab to configure the connection.

**SSL connections.** Both the MySQL server and the client must be configured to enable SSL encryption (see Using Encrypted Connections). To configure this type of connection, set the **Connection Method option** to **TCP/IP (standard)** and use the **Parameters** tab to configure the basic connection. Next, select the **SSL** tab to identify the appropriate files. MySQL Server uses the PEM format for certificates and private keys. In addition to providing the paths to certificate files, you can specify the SSL mode to use for your connection. The following table describes each **Use SSL** option value and indicates which files are required.

<table>
<thead>
<tr>
<th>Option value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No</strong></td>
<td>Do not use SSL. No SSL files are required.</td>
</tr>
<tr>
<td><strong>If Available</strong></td>
<td>Use SSL if the server supports it, but allow connection in all cases. No SSL files are required; however, providing the SSL CA file is the best practice for connections made to MySQL 8.0 servers.</td>
</tr>
<tr>
<td><strong>Require</strong></td>
<td>Always use SSL and deny a connection if the server does not support SSL. Do not perform server certificate validation. No SSL files are required.</td>
</tr>
<tr>
<td><strong>Require and Verify CA</strong></td>
<td>Always use SSL. Validate the certificate authorities (CA), but tolerate a name mismatch. Requires the SSL CA file.</td>
</tr>
</tbody>
</table>
New Schema Creation in Excel

<table>
<thead>
<tr>
<th>Option value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Require and Verify Identity</td>
<td>Always use SSL and fail if the host name is not correct. Requires valid SSL CA, SSL Cert, and SSL Key files.</td>
</tr>
</tbody>
</table>

**SSH connections.** SSH tunnels permit you to connect to a MySQL database from behind a firewall when the MySQL server port is blocked. To configure this type of connection, set the **Connection Method** option to **Standard TCP/IP over SSH** and use the **Parameters** tab to configure the connection.

**Additional considerations:**

- The SSH server communicates with a MySQL server instance in an unencrypted or encrypted mode, based on the value selected for the SSL mode (see Table 6.1, “Use SSL Option Values”). Select the **SSL** tab to specify a value for the **Use SSL** option.
- The account you use to run MySQL for Excel must be authorized on the SSH server in advance. To authenticate your account (specified by **SSH User Name**), you can use one of the following strategies:
  - **SSH User Name** and **SSH Password** only
  - **SSH User Name** and **SSH Key File** only
    Your SSH server may require that you to also provide the **SSH Passphrase** property when using a key file. An invalid pass phrase generates an exception.
  - **SSH User Name** and **SSH Key File (SSH Passphrase)** and **SSH Password**
    The combination of user name + key file (passphrase) + password can perform fallback authentication when the key file and pass phrase are valid, but an error occurs on the server. Specifically, the first attempt to connect uses the key file, and if it fails, the next attempt to connect uses the password instead. If the SSH key file is null or empty, but the SSH password is provided, MySQL for Excel attempts to connect using the SSH password only.

**Editing MySQL Connections**

You can use MySQL for Excel or MySQL Workbench to edit existing MySQL connections.

- Editing MySQL Connections in MySQL for Excel
  To edit a MySQL connection, right-click the connection you want to modify and select **Edit Connection** from the context menu.
  The MySQL connection edit dialog is similar to the edit dialog in MySQL Workbench. Configure the changes and click **OK** to save your changes.

- Editing MySQL Connections in MySQL Workbench
  Alternatively, you can edit your MySQL for Excel MySQL connections using MySQL Workbench. To do this, open MySQL Workbench, edit a MySQL connection, and then refresh the connection list in MySQL for Excel.
  For information about editing MySQL connections in MySQL Workbench, see Connections in MySQL Workbench.

**6.3.2 New Schema Creation in Excel**

A schema contains database objects, such as tables, views, and procedures. You can add a new and empty MySQL schema directly from the MySQL for Excel task pane and then export data from your Excel worksheet to the new schema in MySQL.
To add an empty MySQL schema

1. Start Excel, select the Data menu tab, and then click MySQL for Excel to open the MySQL for Excel task pane.

2. From the Open a MySQL Connection area in the task pane, double-click an existing local or remote connection to display the available database schemas.

3. Click Create New Schema to open the New Schema Name dialog (see the figure that follows).

Figure 6.4 MySQL for Excel: Create a New Schema

To create a new schema:

- In the Schema Name field, type the name of the new schema and then select a collation from the list or keep the server default collation.

- When you click OK, the new schema appears in the list of schemas.

6.3.3 Global Options

Each operation, such as Import MySQL Data, has its own set of options. This section describes the options that apply to all connections, SQL queries, spatial data, imported tables, and edit sessions.

To configure global options, open an existing connection and then click Options in the MySQL for Excel task pane.

Option buttons include:

- Accept: Saves option changes to your host and preserves these changes across all sessions and future Excel instances.

- Reset to Defaults: Resets all option values in the current options window to their default settings. Click Accept to save the changes.

A set of global options affect the entire add-in, as described in the sections that follow.

- Connections Tab
- SQL Queries Tab
- Spatial Data Tab
- Imported Tables Tab
- Edit Sessions Tab
- Manage Import/Edit Connections Information
Connections Tab

The following figure shows the options that apply to all MySQL connections.

Figure 6.5 Global Options: Connections

Connection Options:

- **Wait [ ] seconds for a connection to the server before timing out.** Defaults to 15.
- **Wait [ ] seconds for a database query to execute before timing out.** Defaults to 60.
- **Automatic connections migration delayed until:** Migration status

**Migrate stored connections to MySQL Workbench now**

You can create new MySQL connections in MySQL for Excel when MySQL Workbench is not installed. As soon as you install MySQL Workbench, MySQL for Excel attempts to migrate the saved connections to merge them with those of MySQL Workbench to enable connection reused in both products. MySQL connections cannot be migrated if MySQL Workbench is open.

MySQL for Excel prompts you to either migrate the connections automatically, keep the connections stored with MySQL for Excel only, or to delay the migration by:

- one hour
- one day
- one week
- one month
- indefinitely

Delaying by an hour, day, week, or month displays the prompt to migrate again after that time elapses. If you choose to delay the migration indefinitely, you can click **Migrate stored connections to MySQL Workbench now** to migrate all the new connections from MySQL for Excel to MySQL Workbench manually, as long as MySQL Workbench is installed.

SQL Queries Tab

The following figure shows the options that apply to all SQL queries.
Figure 6.6 Global Options: SQL Queries

SQL Queries Options:

- **Do not show SQL statements sent to the server**: When the option is selected, SQL statements are not displayed and only their results are displayed in the information dialog. Selected by default.

- **Preview SQL statements before they are sent to the server**: When selected, the option adds an extra step to the Create New Schema, Export Data, Append Data, and Edit Data operations before a statement is committed to the server. This option opens the Review SQL Script dialog, as shown in the figure that follows for an export-data operation.

Figure 6.7 The MySQL for Excel configuration: Preview Option

From here you can modify the SQL statements before they are executed, which also enables the **Original Query** button. If clicked, it will revert all modifications to the script to restore the SQL to its original form (when the dialog first opened).

This option is deselected by default.

- **Show executed SQL statements along with their results**: When enabled, SQL statements are first executed and then the information dialog includes both the results and the executed statements. This is helpful when reviewing the recently executed queries when comparing the results.
Global Options

This option is deselected by default.

Spatial Data Tab

The spatial data option (see the figure that follows) enables you to choose a format for handling the data as text. Use the drop-down list to select a format to apply:

- Well-Known Text (default)
- Keyhole Markup Language
- Geography Markup Language
- GeoJSON

Figure 6.8 Global Options: Spatial Data

Imported Tables Tab

The following figure shows the options apply to all imported tables.

Figure 6.9 Global Options: Imported Tables

Imported Tables Options:

- Restore imported MySQL data in Excel Tables when:
  - Opening an Excel workbook: Selected by default.
  - Showing the Add-In’s sidebar: Deselected by default.
Global Options

• Manage connection information stored in the user settings file

Provides a list of saved Excel files with information about MySQL connections. Clicking this button opens the Manage Import/Export Connection Information dialog.

Edit Sessions Tab

The following figure shows the options that apply to all edit sessions.

Figure 6.10 Global Options: Edit Sessions

Edit Session Options:

• Preview MySQL table data before an Edit Data session is opened: Selected by default.

• Use optimistic updates on all Edit Data sessions. This option helps prevent unintentional data overwrite, in that it checks for external edits before committing your changes. For example, between the time you loaded the data into Excel, made changes in Excel, and committed, a different user could have edited the same cells elsewhere in MySQL using MySQL Workbench or some other means. The optimistic updates feature checks for these changes, and notifies you accordingly.

Optimistic updates can also be configured at runtime for all edit sessions, or for a specific edit session by right-clicking the Edit Session floating dialog and choosing the desired Use Optimistic Update option, as shown in the figure that follows.

Figure 6.11 Optimistic Updates: Configuring at Runtime

The optimistic updates option is selected by default.

Tolerance for FLOAT and DOUBLE comparisons in WHERE clause: This related option provides a way to edit data of type FLOAT or DOUBLE that enables proper row-matching in the database when it is used together with optimistic updates. For example, the WHERE clause for optimistic updates uses a construct similar to the following to match rows in the database:

??float_or_double_column?? BETWEEN
• **Restore saved Edit sessions when opening an Excel workbook:** Selected by default.

• **Reuse Excel worksheets matching their names with the session table names:** Selected by default.

• **Create new Excel worksheets for the restored Edit sessions:** Deselected by default.

• **Manage connection information stored in the user settings file**

  Provides a list of saved Excel files with information about MySQL connections. Clicking this button opens the Manage Import/Export Connection Information dialog.

### Manage Import/Edit Connections Information

This dialog lists the connection information for Excel worksheets that are known to MySQL for Excel. From here you can view all saved workbooks that contain worksheets with imported or edited data from MySQL tables and optionally delete the connection information associated with each workbook.

#### Note

The list of missing information will be empty in most cases because MySQL for Excel manages orphaned connection information for you. If you see items in the Manage Import/Edit Connections Information dialog, it means that you should remove connection information as described in this section.

By default, clicking **Accept** deletes connection information for missing worksheets, but this behavior is configurable with the **Delete automatically connection information where Workbook is no longer found** option. In addition, **Select connection information entries** selects (for deletion) the check box of each book you have not accessed for \( n \) days, where \( n \) defaults to 30. The following figure shows this dialog populated with connections.

**Figure 6.12 MySQL for Excel: Manage Connection Information**

![Manage Import/Edit Connections Information](76)
What is New in MySQL for Excel

6.4 What is New in MySQL for Excel

This section summarizes how MySQL for Excel progressed with each minor and major release.

6.4.1 What is New in MySQL for Excel 1.3

Most of the new features added to MySQL for Excel 1.3.x involve improvements to the Data Import functionality.

- Starting with MySQL for Excel 1.3.8, support for secure connections was added. Encrypted connections can be in the form of SSL certificates or SSH tunneling, without the requirement of having intermediate proxy software to create the tunnel. Encrypted connections can be configured from the MySQL for Excel add-in directly or they can be configured with MySQL Workbench and then used to open a connection from the add-in.

- A new global option, Tolerance for FLOAT and DOUBLE comparisons in WHERE clause, provides a way to edit data of type FLOAT and DOUBLE that enables proper row-matching in the database when it is used together with optimistic updates (see Global Options, Edit Sessions Tab).

- The Import Data operation for stored procedures now enables the selection of individual columns to be imported from each returned result set, which is similar to the way imported column data already works for table and view data.

- The Import Data operation adds digits to floating-point numbers. For example, instead of rendering a value such as 5.3 precisely from the database, the operation displays 5.0000019073486 after importing the data. This behavior affects FLOAT and DOUBLE data types, which adhere to the IEEE-754 standard and are stored as approximate values.

A new option now provides a way to import floating-point numbers using the DECIMAL data type, which then stores and displays the exact value from the database (see Advanced Import Data Options, Formatting Tab).

- Support for MySQL Fabric was removed. Fabric connections created in MySQL Workbench will no longer appear in the MySQL for Excel task pane.

- The way MySQL for Excel shares data-editing sessions among users and between computers has been improved to promote maximum portability. A workbook with Excel tables containing MySQL data when saved by one Windows account to a local computer can now be sent to a coworker (different Windows account), or simply moved to a different computer, and the MySQL data can be refreshed and any edit-data sessions can be restored automatically when the workbook is opened.

This new functionality requires the Excel workbook to support XML parts, which is the case for the following file formats:

- Excel Workbook (.xlsx)
- Excel Macro-Enabled Workbook (*.xlsm)
- Excel Binary Workbook (*.xlsb)
- Excel 97-2003 Workbook (*.xls)
What is New in MySQL for Excel 1.3

- Excel Template (*.xltx)
- Excel Macro-Enabled Template (*.xltm)
- Excel 97-2003 Template (*.xlt)
- Microsoft Excel 5.0/95 Workbook (*.xls)
- Strict Open XML Spreadsheet (*.xlsx)

Global options were rearranged into tabs for easier navigation. Click **Options** in the MySQL for Excel task pane to open the Global Options dialog.

- Added a new global option to format spatial data. Formats are: Well-Known Text, Keyhole Markup Language, Geography Markup Language, or GeoJSON.

- Added a new global option (**Imported Tables** tab) to restore imported MySQL data in Excel tables when either the Excel workbook is opened or the MySQL for Excel task pane is opened.

- Added the **Reset to auto-detected** action to the context menu of the **Data Type** field within the **Column Options** area in the Export Data to MySQL dialog.

- The **Connections Migration** dialog now offers a way to postpone the migration of connections to MySQL Workbench by one hour, one day, one week, one month, or indefinitely. For more information about this option, see **Connections Tab**.

- The **Append Excel Data to Table** operation was updated with new advanced options to manage the behavior of rows containing unique key values that are duplicates of those in the database. The following new options are located within the **SQL Queries** tab of the Advanced Append Data Options dialog:
  - **Error out and abort the append operation** (default)
  - **Ignore rows with duplicate unique key values**
  - **Replace the values in the old rows with the ones in new rows**

- You can now refresh imported data from the source MySQL database by clicking **Refresh** from the context-menu, or **Refresh All** from the navigation menu. These actions check for changes in the source MySQL database and update your imported MySQL data accordingly.

  Use case: A colleague sends you a MySQL Excel worksheet with data exported from a MySQL database. You open the file several days later, and worry that the data is outdated so you click **Refresh**.

- A new **Refresh To Defaults** button was added to the options pages. It changes each option to the default value, and you then confirm (or cancel) the application of these changes.

- Enabling the new **Add Summary Fields for Numeric Columns** option adds a summary field to the end of each numeric column in Excel. From here, you choose the desired function for the column, such as total or average.

- You may now import data from multiple objects in a single operation. Use **Control** or **Shift** to select multiple objects (tables and/or views) from the MySQL for Excel task pane, and click **Import** to open the new dialog for selecting additional objects that have direct relationships to the objects you selected. Each object opens in its own Worksheet.

  From this new dialog, you may also generate a **Relationships** model in Excel. This functionality requires Excel 2013 or higher, or Excel 2010 with the PowerPivot add-in.

- A new **Create a PivotTable with the Imported Data** option was added. This creates a Pivot Table in Excel.
• All options now have descriptive tooltips. Hover over an option/preference to view helpful information about its use.

• You may now specify a collation for created schemas. The collation type defaults to "Server Default." These statements can be reviewed before execution.

• All MySQL data types are now available when performing Data Export operations. By default, only the most commonly used data types are listed, which was only behavior in previous versions of MySQL for Excel. You may still type in a type instead of selecting it.

6.4.2 What is New in MySQL for Excel 1.2

• **Edit Connections**: MySQL connections can now be edited from within the MySQL for Excel plugin by right-clicking and choosing Edit Connection. Before, these connections could only be edited with MySQL Workbench.

• **Optimistic Updates**: Previously, only "Pessimistic Updates" were used, which means that pressing Commit Changes would overwrite changes performed outside of MySQL for Excel for the edited cells.

Both options remain available today, and optimistic updates are enabled by default. This update type can be set either as a preference, or toggled per session.

• The **Append Data** dialog will now notify you of incompatible types (with visual warnings) when mapping source Excel columns to target MySQL columns.

If a mismatch is discovered, then the column in the source grid that contains the mapped Excel data turns red, and selecting this column displays a warning with text explaining that the source data is not suitable for the mapped target column's data type.

• New preview preferences allow you to enable one of the following three options:

  • **Preview SQL statements before they are sent to the Server**: View (and optionally) edit the MySQL UPDATE/INSERT statements before they are committed.

  • **Show executed SQL statements along with the results**: View the statements after they are committed, which is the current behavior.

  • **Do not show the MySQL statements**: Only show summary information, such as number of affected rows, and not MySQL statements. This is enabled by default.

• **Create Table**: The Data Export feature now has the option to only create the table without inserting the data.

To execute, toggle the Export Data button to Create Table, and then click.

• The selected schema name is now displayed on top of the MySQL for Excel Database Object Selection window.

• The **Advanced Options** dialogs opened from the Import, Export and Append Data windows now immediately apply the option changes, when before the Advanced Options dialog had to be reopened before the changes could be previewed.

• **Edit Data** sessions can now be saved: Using the new Edit Session preferences, these sessions were automatically closed after closing an Excel workbook. This data, such as the Workbench connection ID, MySQL schema, and MySQL table name, can now be preserved if the Excel workbook is saved to disk, and available when the Excel workbook is reopened.

• Excel tables are automatically created for any data imported from MySQL to an Excel worksheet, with a name like "Schema.DB-Object-name". The DB object name can be a MySQL table, view, or stored procedure. Options for this feature are listed under Import Data, Advanced Options. The newly created Excel tables can be referenced for data analysis in Pivot Tables or reports.
6.5 Edit MySQL Data in Excel

MySQL for Excel simplifies the tasks of inserting, deleting, and updating database table rows using Microsoft Excel by providing a snapshot of the selected MySQL table that is imported into an Excel worksheet. The edit-session feature works with that detached copy of the data. Any changes made to that copy are translated into SQL statements that are applied to the selected MySQL table.

6.5.1 Edit Session Overview

MySQL for Excel enables you to load and edit MySQL table data directly from a Microsoft Excel worksheet. Changes are committed immediately if the Auto-Commit option is enabled in the pop-up menu; otherwise, you can use the Revert Data or Commit Changes operations in the same pop-up menu to revert or commit data changes in MySQL tables.

To edit MySQL table data from an Excel worksheet

1. Start Excel, select the Data menu tab, and then click MySQL for Excel to open the MySQL for Excel task pane.
2. From the Open a MySQL Connection area in the task pane, double-click an existing local or remote connection to display the available database schemas.
3. Select a schema from the list and click Next to display all database objects in the schema (tables, views, and procedures).
4. Select the table with data to edit and then click Edit MySQL Data. A preview window displays the selected data and provides an option to specify the number of rows to preview.
5. Click OK to move the preview data to the current worksheet and to enable edit mode. See Figure 6.13, “Editing table data with MySQL for Excel” for an example of the edit-mode menu (pop-up).

In edit mode, edit the data in your worksheet and click Commit Changes to add the changes manually or select Auto-Commit to commit the changes as you edit the data. Click Revert Data to either refresh data from the database or to revert the changed data.

Additional edit-mode actions:

- To stop edit mode, right-click the pop-up menu and select Exit Edit Mode.
- To resume edit mode, click a cell with data.
- To move the pop-up menu, hold down the right mouse button and drag the menu.

The following figure shows an example of the category table of the sakila schema sample in edit mode. The background color represents the status of each cell. There are four distinct colors that are used while editing table data and two helper colors. For details, see the key of cell colors in edit mode.
In the previous example:

- The green Drama cell was changed and then committed.
- The blue Gaming cell was changed but not committed.
- Finally, **Auto-Commit** was enabled before changing the value 9 to 10 in row 10 of the category_id column, which generated an error because this commit would have added a duplicate value as primary key.

### Table 6.2 Key of Cell Colors in Edit Mode

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
<td>Default color for all cells. This is either the original data, or the data after <strong>Refresh from DB</strong> is clicked.</td>
</tr>
<tr>
<td>Green</td>
<td>Cells that were committed with success.</td>
</tr>
<tr>
<td>Blue</td>
<td>Cells that were modified but have not yet been committed.</td>
</tr>
<tr>
<td>Red</td>
<td>Cells that generated an error when a commit was attempted. An error dialog is also displayed while the commit is attempted.</td>
</tr>
<tr>
<td>Orange</td>
<td>Cells that had a commit attempted, but the commit failed due to detected changes from external sources. For example, a different user made a change to a field after it was imported into Excel. This is a feature of <strong>Optimistic Updates</strong>.</td>
</tr>
<tr>
<td>Yellow</td>
<td>Cells that accept new data. Data entered here is inserted into the MySQL table.</td>
</tr>
</tbody>
</table>
6.5.2 Multiple Edit Sessions

By design, MySQL for Excel does not lock a table when you open an edit session and the table data is imported. Instead, the edit session feature relies on a model in which you have your own copy of the table data (a snapshot) to modify. The model also implies that your coworkers also have their own copy of the data. When multiple edit sessions perform operations on the same table at the same time, it is possible to overwrite the changes made by another session.

To prevent unintended overwrites when multiple edit sessions are in use, be sure the Use optimistic updates on all Edit Data sessions option is selected (see Edit Sessions Tab). This option is selected by default and should not be deselected when the MySQL database is on a remote host that does not guarantee exclusive access of data to you.

The optimistic updates feature works by ensuring that the WHERE clause of the UPDATE statements reference all columns in the table. If any value of a record was changed by another session, while you are also making changes, the record is not found and that UPDATE statement will do nothing when you commit the changes to the database. In contrast, with optimistic updates deselected, the WHERE clause of the UPDATE statements reference just the primary key columns of the table.

Tip
To view the statements generated by MySQL for Excel, select the Preview SQL statements before they are sent to the server option before beginning your edit session (see SQL Queries Tab).

With optimistic updates enabled, non-conflicting changes are committed to the database, but the conflicting changes are not committed. To fix the conflicts, do the following:

1. In your worksheet while still in edit mode, locate the orange-colored cells that show the conflicting data.
2. Retrieve a fresh copy of data from the database by clicking Revert Data on the edit session dialog then Refresh Data from DB.
3. Apply changes to the conflicting cells again and then commit changes.

For general information about working with edit sessions, see Section 6.5.1, “Edit Session Overview”.

6.6 Import MySQL Data into Excel

Data can be imported from MySQL into a Microsoft Excel worksheet by using the Import MySQL Data option after selecting either a table, view, or procedure to import.

Important
The import-data operation ignores columns containing binary data (Varbinary or Blob). For details, see Known limitation for binary data.

To import MySQL data into an Excel worksheet

1. Start Excel, select the Data menu tab, and then click MySQL for Excel to open the MySQL for Excel task pane.
2. From the Open a MySQL Connection area in the task pane, double-click an existing local or remote connection to display the available database schemas.
3. Select a schema from the list and click Next to display all database objects in the schema (tables, views, and procedures).
4. Select the table, view, or procedure with data to import and then click Import MySQL Data. A preview window displays the selected data and provides Options and Advanced Options to be used during the import operation.
For a description of each import option, see Importing a Table or View or Section 6.6.1, “Importing a Procedure”.

For a description of each advanced option, see Section 6.6.2, “Import — Advanced Options”.

5. Click Import to finish the operation.

The Import Data windows provides a preview of the columns to select during the import operation. You can specify both the columns and rows to import. As the following figure shows, the preview includes a small subset of the rows for the selected table or view.

Figure 6.14 Importing table data with MySQL for Excel

Importing a Table or View. Import-data options apply to the current operation only. The options for importing a table or view are:

- Include Column Names as Headers: Selected by default, this inserts the column names at the top of the Microsoft Excel worksheet as a "headers" row.

- Limit to ___ Rows and Start with Row ___: Deselected by default, this limits the range of imported data. The Limit to option defaults to 1, and defines the number of rows to import. The Start with Row option defaults to 1 (the first row), and defines where the import begins. Each option has a maximum value of COUNT(rows) in the table.

- Create a PivotTable with the imported data: Deselected by default. For usage instructions, see Section 6.6.4, “Creating PivotTables”

- Add Summary Fields: Deselected by default, this option adds a summary field to each column. For additional information, see Section 6.6.3, “Adding Summary Fields”.

Choosing Columns To Import. By default, all columns are selected and will be imported. Specific columns may be selected (or unselected) using the standard Microsoft Windows method of either Control + Mouse click to toggle the selection of individual columns, or Shift + Mouse click to select a range of columns.
The background color of a column shows the status of each column. The color white means that the column has been selected, and therefore it will be imported. Conversely, a gray background means that the column will not be imported.

Right-clicking anywhere in the preview grid opens a context-menu with either a Select None or Select All option, depending on the current status.

6.6.1 Importing a Procedure

To import a stored routine (procedure), first select the appropriate routine parameters, if needed, and then click Call. Next, use the following options to configure the import operation:

- **Include Column Names as Headers**: Selected by default, this will insert the column names at the top of the Excel worksheet as a "headers" row.

- **Import**: Because a procedure might return multiple result sets, the import options include:
  - **Selected Result Set**: Imports the selected tab sheet. This is the default behavior.
  - **All Result Sets - Arranged Horizontally**: Imports all result sets into the Excel Worksheet horizontally, and inserts one empty column between each result set.
  - **All Result Sets - Arranged Vertically**: Imports all result sets into the Excel Worksheet vertically, and inserts one empty row between each result set.

For example, a preview area within the Import Data window, similar to the one shown in the following figure, will display after you select routine parameters and then click Call to execute the routine.

**Figure 6.15 Importing called stored procedure data with MySQL for Excel**
6.6.2 Import — Advanced Options

Advanced options enable you to manage the way MySQL for Excel imports data from MySQL tables, views, or procedures. Use Reset to Defaults to restore the original option selections and values.

A set of advanced options affect the current import-data operation, as described in the sections that follow.

- General Tab
- Formatting Tab
- Excel Table Tab

General Tab

The following figure shows the general options that apply to import-data operations.

Figure 6.16 Importing table data with MySQL for Excel: General Options

General Options:

- Use the first [10] rows to preview the MySQL table's data. This option affects the preview step in the import and edit processes and defaults to 10 rows. The value range is 1 to 100.

- Escape text values that start with "=" so Excel does not treat them as formulas. This option may not reflect any differences in the preview because it is only applied after the data is imported into the Excel worksheet.

Formatting Tab

The following figure shows the formatting options that apply to import-data operations.
Figure 6.17 Importing table data with MySQL for Excel: Formatting Options

Formatting Options:

- Excel number format for DateTime and TimeStamp data: \[ m/d/yyyy h:mm \]
- Excel number format for Date data: \[ m/d/yyyy \]
- Excel number format for Time data: \[ hh:mm:ss \]
- Import all floating-point data using a DECIMAL data type. This option imports floating-point data as type DECIMAL, instead of FLOAT or DOUBLE, to show the exact value of data stored the database. When this option is deselected, the floating-point values shown in the preview are approximate values. Selected by default.

Excel Table Tab

The following figure shows the Excel table options that apply to import-data operations.

Figure 6.18 Importing table data with MySQL for Excel: Excel Tables Options

Excel Table Options

- Create an Excel table for the imported MySQL table data. Selected by default and has the following settings:
  
  - Use style [ ] for the new Excel table. Defaults to MySqlDefault.
  
  - Prefix Excel tables with the following text: _______. Deselected by default.
6.6.3 Adding Summary Fields

Summary fields are calculated fields added to the last row of each column in the imported MySQL table. The drop-down list for each summary field in the resulting worksheet includes functions such as Average, Sum, Min, and Max.

Note

This feature was added in MySQL for Excel 1.3.0.

The Add Summary Fields option (deselected by default) appears on the Import Data dialog as shown in the figure that follows.

Figure 6.19 The 'Add Summary Fields' option

![Image of Import Data dialog with 'Add Summary Fields' option highlighted]

Enabling this option adds a row of summary fields for the appropriate columns in your imported data. Notice the newly created row appended to the existing rows shown in the next figure.
6.6.4 Creating PivotTables

A PivotTable can be created from imported MySQL tables, views, stored procedures, or the entire Excel Data Model.

Note
This feature was added in MySQL for Excel 1.3.0.

An Excel PivotTable report summarizes and provides a visual representation of data in many different ways. It is a native Excel feature. For instructions on how to use PivotTables, see the documentation provided by Microsoft.

The following example shows a simple use case where an empty PivotTable is created from an imported MySQL table. This example uses the film table of the sakila database. To create the PivotTable, select the film table from the task pane and then click Import MySQL Data. In the Import Data dialog, select the Create a PivotTable check box and click OK to execute the operation.
Figure 6.22 Option: Create a PivotTable with the imported data

When the *Create a PivotTable with the imported data* option is checked (see the previous figure), an empty PivotTable (or a PivotTable placeholder) is inserted just to the right of the imported data. The PivotTable name follows the same naming rules used for Excel tables created from the imported data, but PivotTables can be created with or without enabling the *Create an Excel table for the imported MySQL data* advanced option. That means a PivotTable can be created from an imported Excel range (if the aforementioned advanced option is off), or from an imported Excel table (if the option is on).

Click *Import* to dump the film table data to an active Excel worksheet, and this also creates a PivotTable for that data as shown in the follow figure.
Clicking the PivotTable opens a PivotTable Fields window next to the MySQL for Excel task pane, and from here you can select fields you want to summarize in the PivotTable report. Drag and drop fields from the list to any of the FILTERS, COLUMNS, ROWS, or VALUES areas, depending on the visualizations you want in the report. The report is completely dynamic, meaning that you can change the views by moving fields around the areas until you see the visualization you need for your PivotTable report.

The next figure shows an example PivotTable report using the sakila.film table imported in the previous example. This report includes a filter by release_year and it summarizes the rental_rate values while also grouping the data by values in the rating column.
Creating PivotTables

Figure 6.24 PivotTable Example: Film Ratings

Expanding one of the groups reveals its values from the title and description columns as shown in the figure that follows.

Figure 6.25 PivotTable Example: Expanded Group

The same operation can be performed with data coming from a MySQL view or stored procedures. One difference for stored procedures enables you to create a PivotTable for each of the imported result sets returned by the procedure call.

In the following figure, the film_in_stock stored procedure is selected and its input parameter values are configured. When the procedure is called, it returns one result set (Result1) and the OutAndReturnValues table (always present if the procedure has output parameters or a return value).
In the next figure, the **All Result Sets - Arranged Horizontally** option was selected. Because the **Create a PivotTable with the imported data** option was also checked, a PivotTable was created for each returned result set.
An important use case for PivotTables is when we create it for multiple related tables as typically a single table does not contain all of the data needed by a PivotTables report. You can create a single PivotTable tied to the data in the current Excel Data Model that contains fields from several related tables. That way you can use the data in a single report for an entire MySQL schema if needed. However, you can only do this in Excel 2013 (and later) where the Excel Data Model is available.

In Excel versions before Excel 2013, only a PivotTable for each imported table or view can be created. This is because a single PivotTable for the entire Excel Data Model requires that the tables are related to each other. If Excel relationships cannot be created, then this type of PivotTable cannot be created. In these cases, the Import Data dialog looks similar to the figure that follows.
Figure 6.28 Disabled Create Excel relationships option before Excel 2013

Clicking Why is this option disabled? displays an information dialog (see the next figure) with an explanation of the disabled controls.

Figure 6.29 Disabled Create Excel relationships option description

The next figure shows an example that uses all tables in the schema. You can choose each table manually or use Control + A in the database objects list to select them all. When clicking Import Multiple Tables and Views, the Import Data dialog appears as shown in the example. Confirm that the Create a PivotTable check box is selected and is set to for all the tables in the data model. Keep that value.
Creating PivotTables

Figure 6.30 Importing All Tables and Views

When clicking **Import**, the data in all of the selected tables are imported to Excel, the Data Model and Excel relationships are created, and a new worksheet is created that contains a PivotTable with all of the tables that were imported. This combination is demonstrated in the figure that follows. Note that all tables are listed in the **PivotTable Fields** window.
You can also configure the Create a PivotTable drop-down list the for each imported table or view (see the next figure), which in turn creates a PivotTable for each of the imported tables or views, as opposed to creating a single PivotTable for all of them.
6.7 Append Excel Data into MySQL

Data from a Microsoft Excel worksheet can be appended to a MySQL database table by using the **Append Excel Data to Table** operation. The sections that follow describe how to preview and map data from a worksheet to a MySQL table.

### 6.7.1 Column Mappings

Mapping the Excel columns to the MySQL columns can be executed automatically (default), manually, or by using a stored mapping routine. An automatic mapping routine is the default, and can be tweaked if every column cannot be matched automatically. The following figure shows several columns of Excel data selected in the original worksheet and the Append Data (preview) dialog after clicking **Append Excel Data to Table**.
General Mapping Information

It is common to tweak the column mappings. A few notes about the manual mapping process:

- Manual mapping is performed by dragging a column from the upper source grid (Excel worksheet) and dropping it into the lower target column MySQL table grid. Click anywhere within the column to initiate this dragging routine.

- The color of the header field for each column defines the current mapping status of the column. The colors include:
  - **Green**: A source column is mapped to a target column.
  - **Red**: A target column is not mapped.
  - **Gray**: A source column is not mapped.

- A source column may be mapped to multiple target columns, although this action generates a warning dialog.

- Right-clicking a target column shows a context menu with options to either **Remove Column Mapping** for a single column, or to **Clear All Mappings** for all columns. Dragging a target column outside of the grid removes the mapping.
Mapping Methods

The three mapping methods are described as follows:

- **Automatic**: The automatic mapping method attempts to match the Excel source column names with the MySQL target table column names. It is then possible to manually tweak the mapping afterward.

  If the automatic process finds zero columns to match, then a simple 1 to 1 matching routine is attempted. Meaning, SourceColumn #1 to TargetColumn #1, SourceColumn #2 to TargetColumn #2, and so on.

- **Manual**: The source column names are manually dragged (matched) with the target column names. Manual dragging can also be performed after the **Automatic** method is selected.

- **Stored**: Manual mapping styles may be saved using the **Store Mapping** button, which will also prompt for a name and then save it using a "name (dbname.tablename)" naming scheme. The saved mapping style will then be available alongside the **Automatic** and **Manual** options. Stored mappings may be deleted or renamed within the **Advanced Options** dialog.

6.7.2 Append — Advanced Options

Advanced options enable you to manage the way MySQL for Excel appends data from an Excel worksheet to MySQL tables. Use **Reset to Defaults** to restore the original option selections and values.

A set of advanced options affect the current append-data operation, as described in the sections that follow.

- **Columns Mapping Tab**
- **Stored Mappings Tab**
- **Field Data Tab**
- **SQL Queries Tab**

Columns Mapping Tab

The following figure shows the columns mapping options that apply to append-data operations.

Figure 6.34 Appending table data with MySQL for Excel: Column Mapping Options

Column Mapping Options:

- **Perform an automatic mapping when dialog opens**: Automatically attempt to map the target and source when the Append Data dialog is opened. This feature is selected by default.
- **Automatically store the column mapping for the given table**: Stores each mapping routine after clicking Append in the Append Data dialog. The mapping routine is saved using the "tablenameMapping (dbname.tablename)" format. This may also be performed manually by clicking Store Mapping. It is selected by default and this feature was added in MySQL for Excel 1.1.0.

- **Reload stored column mapping for the selected table automatically**: If a stored mapping routine exists that matches all column names in the source grid with the target grid, then it is automatically be loaded. This feature is selected by default and it was added in MySQL for Excel 1.1.0.

- **Confirm column mapping overwriting**: Notifies you before overwriting previously saved column mappings and enables you to accept or decline the action. This feature is deselected by default and the option was added for MySQL for Excel 1.3.7.

### Stored Mappings Tab

The following figure shows the stored mappings options that apply to append-data operations.

**Figure 6.35 Appending table data with MySQL for Excel: Stored Mapping Options**

![Stored Mappings Tab]

The Stored Mappings tab shows a list of saved column mappings that were saved with the Automatically store the column mapping for the given table option (in the Column Mapping tab) or manually by clicking Store Mapping in the Append Data dialog.

### Field Data Tab

The following figure shows the field data options that apply to append-data operations.

**Figure 6.36 Appending table data with MySQL for Excel: Field Data Options**

![Field Data Tab]
Field Data Options:

- **Use the first 100** (default) Excel data rows to preview and calculate data types. This determines the number of rows that the preview displays, and the values that affect the automatic mapping feature.

- **Use formatted values**: The data from Excel is treated as **Text**, **Double**, or **Date**. This is selected by default. When deselected, data is never treated as a **Date** type, so for example, this means that a date would be represented as a number.

- **Show column data types above column names**: When selected, the data type of each column appears above the column name in both the source and target grids. This option is deselected by default.

**SQL Queries Tab**

The following figure shows the SQL queries options that apply to append-data operations.

**Figure 6.37 Appending table data with MySQL for Excel: SQL Queries Options**

SQL Queries Options:

- **Generate an INSERT statement for each data row**: When selected, an **INSERT** statement for each data row being appended is generated in the resulting SQL query sent to the MySQL server. Otherwise, a single **INSERT** statement is generated as a bulk data operation, which performs better than multiple **INSERT** statements. This option is deselected by default.

- **Disable table indexes to speed-up rows insertion**: This option is deselected by default, because you must make sure that if unique indexes are present, that the data mapped to that column does not contain duplicate data. This option was added in MySQL for Excel 1.2.1.

- When new rows contain unique key values that duplicate old rows:
  
  - **Error out and abort the append operation**: (default) Retains the behavior of previous versions of MySQL for Excel in which the server returns an error if any duplicate unique key values are found. This option was added in MySQL for Excel 1.3.7.
  
  - **Ignore rows with duplicate unique key values**: Rows containing duplicate values for unique keys are ignored or skipped. Only the rows without duplicate values are inserted into the database. This option was added in MySQL for Excel 1.3.7.
  
  - **Replace the values in the old rows with the ones in new rows**: Rows containing duplicate values for unique keys replace the values of the corresponding rows in the database. This option was added in MySQL for Excel 1.3.7.
6.8 Export Excel Data into MySQL

Data from a Microsoft Excel worksheet can be exported to a new MySQL database table within an existing schema. For instructions on how to add a new schema from MySQL for Excel, see Section 6.3.2, “New Schema Creation in Excel”.

### Important

The export-data operation ignores columns containing binary data (Varbinary or Blob). For details, see Known limitation for binary data.

### To export Excel worksheet data to a MySQL table

1. Start Excel, select the **Data** menu tab, and then click **MySQL for Excel** to open the MySQL for Excel task pane.

2. From the **Open a MySQL Connection** area in the task pane, double-click an existing local or remote connection to display the available database schemas.

3. Select a schema from the list and click **Next** to display all database objects in the schema (tables, views, and procedures).

4. Select the cells in the Excel worksheet to export and then click **Export Excel Data to New Table**. A preview window displays the selected data and provides **Column Options** and **Advanced Options** to be used during the export operation.

5. In the Export Data to MySQL dialog (see the next figure), do the following:

   a. Specify whether the first row in the preview should be used as the column name. This option is selected by default.

      The **First Row Contains Column Names** check box controls the data type for columns either by defaulting the type to a suitable one for rows 2-n (if checked) or to one suitable for rows 1-n (if unchecked).

   b. Type a unique table name and select the collation to use for the new table.

   c. Choose or create a primary key column for the new table.

   d. Select each column and apply the appropriate options and option values.

   e. Click **Export Data** to create the new table with data.
6.8.1 Column Options

The **Column Options** area is located below the preview grid. When you click each column header of the preview grid, you see the option values for the selected column. Some values are set by default: **Column Name**, **Data Type**, and so on. If a column option does not apply to the selected column or if the value cannot be changed, the option appears dimmed.

Use **Advanced Options** to modify the default behavior of the options that apply to all columns intended for export to a MySQL table.

MySQL for Excel analyzes and detects the data type of each column by default; however, you can override the detected value by selecting a different data type from the drop-down list. To reset a column back to the detected value, right-click within the **Data Type** field and select **Reset to auto-detect**, as shown in the figure that follows.

**Figure 6.39 Reset a Data Type**

6.8.2 Export — Advanced Options

Advanced options enable you to manage the way MySQL for Excel exports data from worksheets to MySQL tables. Use **Reset to Defaults** to restore the original option selections and values.
A set of advanced options affect the current export-data operation, as described in the sections that follow.

- **Column Tab**
- **Field Data Tab**
- **SQL Queries Tab**

**Column Tab**

The following figure shows the column options that apply to export-data operations.

**Figure 6.40 Exporting Excel data to MySQL: Column Options**

Column Options:

- **Use the first 100 (default) Excel data rows to preview and calculate data types**: This determines the number of rows that the preview displays and the values that affect the automatic mapping feature.

- **Analyze and try to detect correct data type based on column field contents**: Attempts to analyze the data and determine the data type for the column. The column type is defined as VARCHAR if it contains multiple types.

- **Add additional buffer to Varchar length (round up to 12, 25, 45, 125, 255)**: When the data type is automatically detected and is set to VARCHAR, then it calculates the maximum length for all rows within the column, and rounds up the maximum length to one of the defined lengths above.

  If deselected, then the VARCHAR length is set to the length of the longest entry in the Excel worksheet.

- **Automatically check the Index checkbox for Integer columns**: If selected (default), columns with an Integer data type will have the **Create Index** option selected by default.

- **Automatically check the Allow Empty checkbox for columns without an index**: If selected (default), columns without the **Create Index** check box selected will automatically enable the **Allow Empty** configuration option.

- **Show all available MySQL data types in the Data Type drop-down list**: By default, only the most commonly used data types are displayed. Select (deselected by default) this setting to see a list of all MySQL data types.
Export — Advanced Options

Note
This option was added in MySQL for Excel 1.3.0

Field Data Tab

The following figure shows the field data options that apply to export-data operations.

**Figure 6.41 Exporting Excel data to MySQL: Field Data Options**

Field Data Options

- **Use formatted values**: When selected (default), the data from Excel is treated as Text, Double, or Date. When deselected, data is never treated as a Date type, so for example this means that a date would be represented as a number.

SQL Queries Tab

The following figure shows the SQL query options apply to export-data operations.

**Figure 6.42 Exporting Excel data to MySQL: SQL Queries Options**

SQL Query Options:
• **Generate an INSERT statement for each data row:** When selected, an INSERT statement for each data row being exported is generated in the resulting SQL query sent to the MySQL server. Otherwise, a single INSERT statement is generated as a bulk data operation, which performs better than multiple INSERT statements. This option is deselected by default.

• **Create table’s indexes after data has been exported to speed-up rows insertion:** This saves disk I/O for bulk inserts (thousands of rows) because re-indexing will not happen every time a row is inserted, but only once at the end of the data insertion. This option is selected by default.

  **Note**
  This option was added in MySQL for Excel 1.2.1.

  **Note**
  The following option was Removed in MySQL for Excel 1.2.1. Now, the default behavior is to always remove empty columns from the calculations.

  Remove columns that contain no data, otherwise flag them as "Excluded": If selected, columns without data in Excel are removed and not shown in the preview window. If deselected (default), these columns will exist but have the Exclude Column option checked.

### 6.8.3 Additional Notes

• Entering "0" into a date column.

  Entering the value "0" into an Excel date column will convert the value to "12/30/1899" in MySQL. This is because Excel first translates the value to the minimum date in Excel, which is "1/0/1900", because dates are internally stored in Excel as numbers (the days that have passed since "1/0/1900". However, because "1/0/1900" is not a valid date, the committed value to MySQL will change to "12/30/1899" because the .NET MySQL connector automatically converts "1/0/1900" to "12/30/1899", which is the closest valid date.

### 6.9 MySQL for Excel Frequently Asked Questions

**Questions**

• **6.9.1:** When I'm using Excel to edit data from a live MySQL database, will my changes overwrite changes made by other users? For example, maybe they used MySQL Workbench to edit the same data at the same time.

• **6.9.2:** I installed the MySQL for Excel plugin, but can't find it in Microsoft Excel. How do I start it?

• **6.9.3:** I click on **Edit Data** and after importing the table data into Excel, I can't sort or move columns around. How can I do that?

• **6.9.4:** When editing a MySQL table's data, the Excel worksheet where the data is dumped is protected. How can unprotect it?

**Questions and Answers**

**6.9.1:** When I'm using Excel to edit data from a live MySQL database, will my changes overwrite changes made by other users? For example, maybe they used MySQL Workbench to edit the same data at the same time.

The optimistic updates feature checks for external edits and notifies you of their existence before committing any changes. If an external edit is discovered, you can then choose whether or not to overwrite their changes. This option is enabled by default and can be disabled (to use pessimistic updates). Disabling this option means external changes will always be overwritten. In other words, the choice is yours.
6.9.2: I installed the MySQL for Excel plugin, but can't find it in Microsoft Excel. How do I start it?

The MySQL for Excel plugin is automatically added to Microsoft Excel's data menu when it is installed. Look for the MySQL for Excel icon, by default it will be listed on the right side of the main menu.

If it's not there, then you might have to reinstall the plugin. But before doing so, first check if it's listed under "Add/Remove Programs" in Microsoft Windows. If not, then it has not been installed. Next, check the Excel Add-Ins list. For Office 2007 this is found by clicking the Office logo in Excel (top left corner), click Excel Options, then select Add-Ins. Is MySQL for Excel listed as a COM Add-in? If so, then consider filing a bug report (bugs.mysql.com), or attempt to reinstall the plugin.

6.9.3: I click on Edit Data and after importing the table data into Excel, I can't sort or move columns around. How can I do that?

In order to maintain the mapping of rows and columns in the Excel Worksheet against the rows and columns in the MySQL table, no alteration is permitted on the worksheet (i.e. sorting, deleting rows, deleting columns). If you need to alter the data there you can do that by right-clicking the Edit Data window and selecting Exit Edit Mode.

6.9.4: When editing a MySQL table's data, the Excel worksheet where the data is dumped is protected. How can unprotect it?

The Excel worksheet is protected to not allow alterations to the order of rows and columns. The password used for the protection is a GUID auto-generated at runtime so that the protection is not violated in any way. If you wish to unprotect the worksheet to manipulate your data, you can do that by right-clicking the Edit Data window and selecting Exit Edit Mode.
Chapter 7 MySQL for Visual Studio

Table of Contents

7.1 General Information .......................................................... 109
  7.1.1 New in Version 2.0 ....................................................... 110
  7.1.2 New in Version 1.2 ..................................................... 114
7.2 Installing MySQL for Visual Studio ...................................... 116
7.3 Enabling the MySQL Toolbar .............................................. 118
7.4 Making a Connection ......................................................... 119
  7.4.1 Connect Using Server Explorer .................................. 121
  7.4.2 Connect Using MySQL Connections Manager .................. 123
7.5 Editing ......................................................................... 125
  7.5.1 MySQL SQL Editor ..................................................... 125
  7.5.2 Code Editors ............................................................ 126
  7.5.3 Editing Tables ........................................................... 128
  7.5.4 Editing Views ........................................................... 133
  7.5.5 Editing Indexes .......................................................... 135
  7.5.6 Editing Foreign Keys .................................................. 136
  7.5.7 Editing Stored Procedures and Functions ...................... 137
  7.5.8 Editing Triggers ........................................................ 139
7.6 MySQL Project Items ........................................................ 140
  7.6.1 MySQL ASP.NET MVC Items ......................................... 140
  7.6.2 MySQL Windows Forms Items ....................................... 147
7.7 MySQL Application Configuration Tool ................................ 149
  7.7.1 Entity Framework ...................................................... 150
  7.7.2 Web Providers .......................................................... 152
  7.7.3 Using the MySQL Connection String Editor .................... 156
7.8 MySQL Data Export Tool .................................................. 157
7.9 DDL T4 Template Macro .................................................... 163
7.10 Debugging Stored Procedures and Functions ......................... 164
7.11 MySQL for Visual Studio Frequently Asked Questions ............ 174

7.1 General Information

This chapter provides general information about MySQL for Visual Studio and how it has changed.

MySQL for Visual Studio provides access to MySQL objects and data from Visual Studio. As a Visual Studio package, MySQL for Visual Studio integrates directly into Server Explorer providing the ability to create new connections and work with MySQL database objects.

Functionality concepts includes:

- **SQL Development**: By integrating directly into Visual Studio, database objects (tables, views, stored routines, triggers, indexes, etc) can be created, altered, or dropped directly inside Server Explorer.

  Visual object editors include helpful information to guide you through the editing process. Standard data views are also available to help you view your data.

- **Query Designer**: Visual Studio's query design tool is also directly supported. With this tool, you can query and view data from tables or views while also combining filters, group conditions, and parameters. Stored routines (both with and without parameters) can also be queried.

- **Stored Routine Debugging**: Use the full debugging support for stored routines. Using the standard Visual Studio environment and controls, you can set breakpoints, add watches, and step into, out of, and over routines and calls. Local variables can be added to the watch window and call stack navigation is also supported.
New in Version 2.0

- **Entity Framework**: The Entity Framework is supported, to allow template based code generation and full support of the model designers and wizards.

For notes detailing the changes in each release, see the MySQL for Visual Studio Release Notes.

### 7.1.1 New in Version 2.0

This section summarizes many of the new features added to the 2.0 release series in relation to the MySQL for Visual Studio 1.2 release series. MySQL for Visual Studio 2.0.5 is a development release.

New features are described in the following sections:

- Viewing MySQL Query Output
- Version Support for Visual Studio
- Switching Connections from Script and Code Editors
- Making a Connection
- MySQL Toolbar
- MySQL JavaScript and Python Code Editors

For notes detailing the changes in each point release, see the MySQL for Visual Studio Release Notes.

#### Viewing MySQL Query Output

An output pane was added to the MySQL SQL, JavaScript, and Python editors to display information about each executed query. The output pane includes the information that previously appeared in the Messages tab.

**Figure 7.1 MySQL SQL Editor Output**

![MySQL SQL Editor Output](image)

#### Version Support for Visual Studio

Beginning with MySQL for Visual Studio 2.0.5:

- Support for Microsoft Visual Studio 2017 was added.
- Support for Microsoft Visual Studio 2010 was removed.

#### Switching Connections from Script and Code Editors

A drop-down list was added to the toolbar of the SQL, JavaScript, and Python editors from which you can select a valid connection. JavaScript and Python editors show only the connections that support the X Protocol.
Making a Connection

A new MySQL Connections Manager tool was added, and it can create and manage MySQL connections. It is found under the Server Explorer.

Figure 7.3 Opening the MySQL Connections Manager Dialog

This button opens the MySQL Connections Manager dialog that enables the sharing of stored MySQL connections with MySQL Workbench, if it is installed. MySQL connections are displayed in a simpler way and can be created and edited from within this dialog. These connections can be imported to the Visual Studio Server Explorer for use with Visual Studio.

After opening the MySQL Connections Manager:
New in Version 2.0

Figure 7.4 MySQL Connections Manager Dialog: Choosing a Connection

To add a new MySQL connection with the **MySQL Connections Manager**: 

Figure 7.5 MySQL Connections Manager Dialog: New Connection

**MySQL Toolbar**

In the **Server Explorer**, and with MySQL Server 5.7, the MySQL connection context-menu was changed to show the options to create JavaScript or Python scripts, along with the existing SQL script option.
Select **JavaScript** or **Python** to launch the MySQL code editor for the selected language.

**MySQL JavaScript and Python Code Editors**

Use the code editor to write and execute JavaScript or Python queries with MySQL Server 5.7 and higher, or as before, use SQL queries.

**Figure 7.7 MySQL Editor: Script Template**

Select **MyJs Script** or **MyPy Script** to launch the MySQL code editor for the selected language.
7.1.2 New in Version 1.2

This section summarizes many of the new features added to 1.2.x in relation to earlier versions of MySQL for Visual Studio.

- Support for MySQL 8.0 Features
New in Version 1.2

- New or Changed Tool Support
- Version Support for Visual Studio
- Item Templates versus Project Templates

For notes detailing the changes in each point release, see the MySQL for Visual Studio Release Notes.

**Support for MySQL 8.0 Features**

- Starting with MySQL for Visual Studio 1.2.9, SSL PEM and SSH connections can be made to MySQL using the classic protocol (3306 default port number). When the server supports SSL connections, PEM and PFX certificates are permitted with Connector/NET 8.0.16 or higher. Standard TCP/IP over SSH connections are also supported and require MySQL Connector/NET 8.0.17 or higher. For configuration instructions, see:
  - SSL Connections with Server Explorer
  - SSH Connections with Server Explorer

- MySQL for Visual Studio 1.2.8 supports the MySQL 8.0 release series (requires MySQL Connector/NET 6.9.12, 6.10.7, or 8.0.11) including:
  - MySQL data dictionary, which uses INFORMATION_SCHEMA tables rather than tables in the mysql database (see MySQL Data Dictionary).
  - The caching_sha2_password authentication plugin introduced in MySQL 8.0 (see Caching SHA-2 Pluggable Authentication).

**New or Changed Tool Support**

- Starting with MySQL for Visual Studio 1.2.9, the plugin detects when the version of Connector/NET has been changed after MySQL for Visual Studio was installed and prompts to update the necessary configuration files using the Configuration Update Tool. Visual Studio must be restarted to activate the updated configuration files.

- The MySQL Website Configuration tool was renamed to MySQL Application Configuration and extended to automate entry updates to the app.config file in the MySQL for Visual Studio 1.2.9 release.

**Version Support for Visual Studio**

- MySQL for Visual Studio 1.2.9:
  - Support for Microsoft Visual Studio 2019 was added.
  - Support for Microsoft Visual Studio 2012 and 2013 was removed.

- MySQL for Visual Studio 1.2.7:
  - Support for Microsoft Visual Studio 2017 was added.
  - Support for Microsoft Visual Studio 2010 was removed.

**Item Templates versus Project Templates**

Beginning with MySQL for Visual Studio 1.2.5, the project templates used to create MySQL Windows Forms and MySQL MVC projects are no longer be available, as they were replaced with MySQL Project Items:

- **MySQL MVC Item** replaces MySQL MVC Project.
- **MySQL Windows Forms Item** replaces Windows Form Project.
These item templates offer the benefit of adding items to existing projects new windows forms or MVC controllers/views connected to MySQL, based on MySQL Entity Framework models, without the need of create an entirely new MySQL project.

In addition, item templates better follow the Visual Studio template standards, which are oriented to create projects regardless of the database connectivity.

For information about using Item Templates, see Section 7.6, “MySQL Project Items”.

7.2 Installing MySQL for Visual Studio

MySQL for Visual Studio is an add-on for Microsoft Visual Studio that simplifies the development of applications using data stored by the MySQL RDBMS. Many MySQL for Visual Studio features also require that MySQL Connector/NET be installed on the same host where you perform Visual Studio development. Connector/NET is a separate product with several versions.

The options for installing MySQL for Visual Studio are:

• Using MySQL Installer (preferred): Download and execute the MySQL Installer.
  
  With this option you can download and install MySQL Server, MySQL for Visual Studio, and Connector/NET together from the same software package, based on the server version. Initially, MySQL Installer assists you by evaluating the software prerequisites needed for the installation. Thereafter, MySQL Installer enables you to keep your installed products updated or to easily add and remove related MySQL products.
  
  For additional information about using MySQL Installer with MySQL products, see Section 1.3, “MySQL Installer for Windows”.

• Using the standalone Zip or MSI file: This option is ideal if you have MySQL Server and Connector/NET installed already. Use the information in this section to determine which version of MySQL for Visual Studio to install.

MySQL for Visual Studio Configuration Update Tool

The MySQL for Visual Studio installation updates its configuration automatically when Connector/NET is installed on your computer. However, if you install Connector/NET after MySQL for Visual Studio, or upgrade Connector/NET after you have installed MySQL for Visual Studio, you are prompted to run the Configuration Update Tool.

A configuration deviation, if present, is detected by MySQL for Visual Studio when you attempt to create an Entity Framework model or the first time you attempt to open, create, or edit a connection to MySQL. Optionally, you can have MySQL for Visual Studio update the configuration files for you when it detects configuration errors and provides a popup window prompting you to invoke the tool. Select Yes to run the tool. You must restart Visual Studio after the affected configuration files are updated.

Minimum Requirements

MySQL for Visual Studio operates with several versions of Visual Studio, although the extent of support is based on your installed versions of Connector/NET and Visual Studio. MySQL for Visual Studio 1.2.9 no longer supports Visual Studio 2013 or 2012.

Minimum requirements for the supported versions of Visual Studio are as follows:

• Visual Studio 2019 (Community, Professional, and Enterprise)

MySQL for Visual Studio 1.2.9 with Connector/NET 8.0.14

Tip

Connector/NET 8.0.18 is recommended.
MySQL Connector/NET Restrictions

- Visual Studio 2017 (Community, Professional, and Enterprise):
  MySQL for Visual Studio 1.2.7 or 2.0.5 with Connector/NET 6.9.8

- Visual Studio 2015 (Community, Professional, and Enterprise):
  MySQL for Visual Studio 1.2.7 or 2.0.2 with Connector/NET 6.9.8

Minimum requirements for the previously supported versions of Visual Studio include:

- Visual Studio 2013 (Professional, Premium, Ultimate):
  - .NET Framework 4.5.2 (install first).
  - MySQL for Visual Studio 1.2.1 or 2.0.0 with Connector/NET 6.9.8

- Visual Studio 2012 (Professional, Test Professional, Premium, Ultimate):
  - .NET Framework 4.5.2 (install first).
  - MySQL for Visual Studio 1.2.1 or 2.0.0 with Connector/NET 6.9.8

MySQL for Visual Studio does not support Express versions of Microsoft development products, including the Visual Studio and the Microsoft Visual Web Developer. To use Connector/NET with Express versions of Microsoft development products, use Connector/NET 6.9 or later, without installing the MySQL for Visual Studio.

The following table shows the support information for MySQL for Visual Studio.

**Table 7.1 Support Information for Companion Products**

<table>
<thead>
<tr>
<th>MySQL for Visual Studio Version</th>
<th>MySQL Connector/NET Version Supported</th>
<th>Visual Studio Version Supported</th>
<th>MySQL Server Versions Supported</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2 (GA)</td>
<td>8.0</td>
<td>2019, 2017, 2015, 2013, 2012</td>
<td>8.0, 5.7, 5.6</td>
<td>Support for MySQL 8.0 features requires MySQL for Visual Studio 1.2.8 or higher.</td>
</tr>
<tr>
<td>2.0 (RC)</td>
<td>8.0, 6.10</td>
<td>2017, 2015, 2013, 2012</td>
<td>5.7, 5.6</td>
<td>Enables MySQL Configurations Manager and code editors (with MySQL 5.7).</td>
</tr>
</tbody>
</table>

**MySQL Connector/NET Restrictions**

MySQL for Visual Studio is closely tied to Connector/NET, but they are two separate products that can be used without one another. The following restrictions apply:

- MySQL for Visual Studio cannot be installed alongside any version of Connector/NET 6.6 and earlier, which must be removed before installing MySQL for Visual Studio.

- The following MySQL for Visual Studio features require Connector/NET:
  - The Entity Framework Designer
  - The Application Configuration tool
  - Debugging Stored Procedures and Functions
• The DDL T4 Template Macro (to generate a database from an EF Model)

7.3 Enabling the MySQL Toolbar

The optional MySQL toolbar includes MySQL specific functionality and links to external MySQL tools such as MySQL Workbench and MySQL Utilities. Additional actions are available from the context menu for each data connection.

After installing MySQL for Visual Studio, the MySQL toolbar is available by selecting View, Toolbars, MySQL from the main menu. To position the MySQL toolbar within Visual Studio, do the following:

1. From the main menu, click Tools and then Customize.
2. In the Toolbars tab, select MySQL to highlight it. The check box should have a check mark to indicate that the toolbar is visible.
3. Select a dock location from Modify Selection. For example, the following figure shows the MySQL toolbar in the Dock location: Left position. Other dock locations are Top, Right, and Bottom.

Figure 7.10 MySQL for Visual Studio Toolbar and Context Menu

The MySQL toolbar provides shortcuts to some of the main features of MySQL for Visual Studio:

• MySQL Script Window: Opens a new MySQL script window using the selected connection. All available MySQL connections are listed in a submenu, which can be selected on the toolbar:

Figure 7.11 The MySQL for Visual Studio Toolbar: Connections
Making a Connection

The MySQL script window supports the IntelliSense feature for easing MySQL script creation inside Visual Studio.

- **Debug MySQL Routine**: Starts a debugging session on a selected MySQL stored routine inside Visual Studio.
- **MySQL Data Export Tool**: Opens a new tabbed-window of the Data Export tool.
- **MySQL Workbench SQL Editor**: Opens a new Workbench with an SQL editor window using the current MySQL connection, if MySQL Workbench has been installed.
- **MySQL Utilities Console**: Opens a new console window for the MySQL Utilities tool, if it is installed.

### 7.4 Making a Connection

MySQL for Visual Studio leverages and extends the connection capabilities of Visual Studio to create and store MySQL connections. To ensure access to the full set of connection options, install the most recent version of MySQL Connector NET on the client computer (see Minimum Requirements).

MySQL for Visual Studio provides the following two distinct release series:

- **MySQL for Visual Studio 1.2 release series**: Provides General Availability (GA) releases for use with MySQL 5.6, 5.7, and 8.0 servers. Connections can be made to MySQL using the classic protocol only (3306 default port number). When the server supports SSL connections, PEM and PFX certificates are permitted with Connector NET 8.0.16 or higher. Standard TCP/IP over SSH connections are also supported and require MySQL Connector NET 8.0.17 or higher. Unencrypted connections are supported with most versions of MySQL Connector NET. To add or modify MySQL data connections, use Server Explorer in Visual Studio (see Section 7.4.1, “Connect Using Server Explorer”).

- **MySQL for Visual Studio 2.0 release series**: Provides a development release series for use with MySQL 5.6 and 5.7 servers (version 2.0.5 is the current milestone). Basic connections are supported for both the classic MySQL protocol and X Protocol (3306 default port number). When the server supports SSL connections, PEM and PFX certificates are permitted with X Protocol; PFX certificates are permitted with the classic MySQL protocol. Use either MySQL Connections Manager or Server Explorer in Visual Studio to add or modify classic MySQL protocol connections. To add or modify X Protocol connections, use MySQL Connections Manager only (see Section 7.4.2, “Connect Using MySQL Connections Manager”).

MySQL Connections Manager was introduced in the MySQL for Visual Studio 2.0 release series to simplify the creation and management of MySQL server connections within Visual Studio. From MySQL Connections Manager, you have the option to migrate all stored connections to MySQL Workbench (if it is installed), making MySQL Workbench a central repository for MySQL connections.

### Basic Connections in Visual Studio

A basic connection is either unencrypted or encrypted (in MySQL 8.0, SSL is enabled by default) and the connection is made using standard TCP/IP, which is the default connection method in MySQL for Visual Studio to connect to the MySQL RDBMS. Basic connections are easy to configure, particularly if the client application and MySQL server are on the same host computer or operate within the same local area network. For instructions on how to create a basic connection to MySQL from within Visual Studio, see Basic Connections with Server Explorer or Basic Connections with MySQL Connections Manager.

### SSL Connections in Visual Studio

MySQL Server uses the PEM format for certificates and private keys. Connector NET 8.0.17 enables the use of either PEM or PFX certificates with the classic MySQL protocol when Server Explorer
SSH Connections in Visual Studio

in Visual Studio (with MySQL for Visual Studio 1.2.9 or higher) is used to add or modify the data connection.

Both the MySQL server and the client must be configured to enable SSL encryption (see Using Encrypted Connections). In addition to providing the paths to certificate files, the client can specify the SSL mode to use for connections. When using Server Explorer, the SSL mode value is set with an advanced property. MySQL Connections Manager provides the Use SSL Encryption drop-down list with similar values. The following table describes the optional SSL values to select (and the files to specify) with each tool.

Table 7.2 SSL Mode Values

<table>
<thead>
<tr>
<th>Server Explorer</th>
<th>Connections Manager</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>No</td>
<td>Do not use SSL. No SSL files are required.</td>
</tr>
<tr>
<td>Preferred</td>
<td>If Available</td>
<td>Use SSL if the server supports it, but allow connection in all cases. Preferred is the default value with Connector/NET 8.0.11 or higher. No SSL files are required; however, providing the SSL CA file (with either a .pem or .pfx file extension) is the best practice for connections made to MySQL 8.0 servers.</td>
</tr>
<tr>
<td>Required</td>
<td>Require</td>
<td>Always use SSL and deny a connection if the server does not support SSL. Do not perform server certificate validation. No SSL files are required.</td>
</tr>
<tr>
<td>VerifyCA</td>
<td>Require and Verify CA</td>
<td>Always use SSL. Validate the certificate authorities (CA), but tolerate a name mismatch. Requires the SSL CA file. Use either a .pem or .pfx file extension.</td>
</tr>
<tr>
<td>VerifyFull</td>
<td>Require and Verify Identity</td>
<td>Always use SSL and fail if the host name is not correct. Requires valid SSL CA, SSL Cert, and SSL Key files for PEM (.pem file extension). Requires the SSL CA file for PFX certificates (.pfx file extension).</td>
</tr>
</tbody>
</table>

SSH Connections in Visual Studio

SSH tunneling (or port forwarding) enables you to create a secure connection between your computer and a remote computer through which you can access MySQL data. SSH tunnels permit you to connect to a MySQL database from behind a firewall when the MySQL server port is blocked.

There are several considerations for connecting with standard TCP/IP over SSH:

• Your application (or the user account making the connection) requires authorization on the SSH server.

• The SSH server communicates with a MySQL server instance in an unencrypted or encrypted mode, based on the value selected for the SSL mode (see Table 7.2, “SSL Mode Values”). The MySQL server instance does not require additional configuration for this type of connection.

• The Connection Protocol property must use the default value (socket) to establish a connection over standard TCP/IP when configured with Server Explorer.

• To authenticate the client requesting a connection (specified by SSH User Name), you can use one of the following strategies:
  - SSH User Name and SSH Password only
  - SSH User Name and SSH Key File only

  Your SSH server may require that you to also provide the SSH Passphrase property when using a key file. An invalid pass phrase generates an exception.
Connect Using Server Explorer

- **SSH User Name** and **SSH Key File (SSH Passphrase)** and **SSH Password**

The combination of user name + key file (passphrase) + password can perform fallback authentication when the key file and pass phrase are valid, but an error occurs on the server. Specifically, the first attempt to connect uses the key file, and if it fails, the next attempt to connect uses the password instead. If the SSH key file is null or empty, but the SSH password is provided, MySQL for Visual Studio attempts to connect using the SSH password only.

### 7.4.1 Connect Using Server Explorer

This section describes how to create a new connection with or without encryption. After a connection is successfully established, all settings are saved for future use. When you start Visual Studio for the next time, open the connection node in Server Explorer to establish a connection to the MySQL server again. The instructions for setting up connections are provided in these sections.

- **Basic Connections with Server Explorer**
- **SSL Connections with Server Explorer**
- **SSH Connections with Server Explorer**

To modify or delete a connection, use the Server Explorer context menu for the corresponding node. You can modify any of the settings by overwriting the existing values with new ones. The connection may be modified or deleted only if no active editor for its objects is opened; otherwise, you may lose your data.

**Basic Connections with Server Explorer**

To create a connection to an existing MySQL database:

1. Start Visual Studio and open the Server Explorer by clicking **Server Explorer** from the **View** menu.
2. Right-click the Data Connections node and then select **Add Connection**.
3. From the Add Connection window, click **Change** to open the Change Data Source dialog box, then do the following:
   a. Select **MySQL Database** from the list of data sources. Alternatively, you can select <other>, if MySQL Database is absent.
   b. Select **.NET Framework Data Provider for MySQL** as the data provider.
   c. Click **OK** to return to the Add Connections window.
4. Type a value for each of the following connection settings:
   - **Server name:**
     - The name or IP address of the computer hosting the MySQL server. For example, **localhost** if the MySQL server is installed on the local computer.
   - **User name:**
     - The name of a valid MySQL database user account.
   - **Password:**
     - The password of the user account specified previously. Optionally, click **Save my password** to avoid having to enter the password in the Modify Connections window for each connection session.
Connect Using Server Explorer

- **Database name:**

  The database to use as the default schema. You can leave the name blank and select a default schema later from the list of schema on the target server.

  You can also set the port to connect with the MySQL server by clicking **Advanced**. To test connection with the MySQL server, set the server host name, the user name and the password, and then click **Test Connection**. If the test succeeds, the success confirmation dialog box opens.

5. Click **OK** to create and store the new connection. The new connection with its tables, views, stored procedures, stored functions, and UDFs now appears within the **Data Connections** node of Server Explorer.

**SSL Connections with Server Explorer**

You can enable SSL encryption for a classic MySQL protocol connection from Server Explorer. Both SSL PEM and PFX certificate formats are permitted. In addition, MySQL Connector/NET version 8.0.17 must be installed on the client host.

To create a connection with SSL encryption enabled:

1. Add and test a new basic connection (see **Basic Connections with Server Explorer**).

   To modify an existing connection, right-click the connection node within **Data Connections** and select **Modify Connection**.

2. In the Add (or Modify) Connection window, click **Advanced** to open the Advanced Properties dialog box. Advanced properties are categorized and presented in a two-column list, showing the property name and value field (or value list). Default values are not shown.

3. In the **Connection** property category, do the following:

   a. Select **Connection Protocol** and then select **Socket** from the value list (use the arrow in the value field to open the list). This property sets the connection protocol to use standard TCP/IP.

   b. Select **Port** and type **3306** in the value field.

4. In the **Authentication** property category, select **Ssl Mode** and choose the type of mode that best represents your connection. For a description of each mode and the required files, see **Table 7.2, “SSL Mode Values”**.

   - For SSL PEM, use the **Ssl CA**, **Ssl Cert**, and **Ssl Key** properties to add the required files (must have a .pem file extension).

   - For SSL PFX (PKCS#12 format), use the **Certificate File**, **Certificate Password**, **Certificate Store Location**, and **Certificate Thumbprint** properties to add the required information or files (must have a .pfx file extension).

   Click **OK** to close the Advanced Properties dialog box.

5. Click **Test Connection** and adjust the property values if needed.

6. Click **OK** to create and store the new or modified connection.

**SSH Connections with Server Explorer**

Standard TCP/IP over SSH is supported for classic MySQL protocol connections only. In addition, MySQL Connector/NET version 8.0.17 must be installed on the client host and the SSH server must be configured in advance.

To create a connection for standard TCP/IP over SSH:
1. Add and test a new basic connection (see Basic Connections with Server Explorer).
   To modify an existing connection, right-click the connection node within Data Connections and
   select Modify Connection.

2. In the Add (or Modify) Connection window, click Advanced to open the Advanced Properties dialog
   box. Advanced properties are categorized and presented in a two-column list, showing the property
   name and value field (or value list). Default values are not shown.

3. In the Connection property category, do the following:
   a. Select Connection Protocol and then select Socket from the value list (use the arrow in the
      value field to open the list). This property sets the connection protocol to use standard TCP/IP.
   b. Select Port and type 3306 in the value field.

4. In the Authentication property category, select Ssl Mode and then choose the type of mode
   that best represents your connection. For a description of each mode and the required files, see
   Table 7.2, “SSL Mode Values”.

5. In the SSH property category, add values to the properties that apply to your connection (see SSH
   Connections in Visual Studio). Click OK to close the Advanced Properties dialog box.

6. Click Test Connection and adjust the property values if needed.

7. Click OK to create and store the new or modified connection.

7.4.2 Connect Using MySQL Connections Manager

This section describes how to create a new connection with or without encryption. After a connection
is successfully established, all settings are saved for future use. When you start Visual Studio for the
next time, open the connection node in Server Explorer to establish a connection to the MySQL server
again. The instructions for setting up connections are provided in these sections.

- Basic Connections with MySQL Connections Manager
- SSL Connections with Connections Manager

To modify or delete a connection, start MySQL Connections Manager and select an existing
connection. You can modify any of the settings by overwriting the existing values with new ones. The
connection may be modified or deleted only if no active editor for its objects is opened; otherwise, you
may lose your data.

Basic Connections with MySQL Connections Manager

To create a connection to an existing MySQL database:

1. Click in the Server Explorer menu bar to open the MySQL Connections Manager window.

2. Click Add New Connection to create a new connection.

3. Provide a unique name for the new connection in the required Connection Name field.

4. Confirm that TCP/IP (standard) is selected as the connection method.

5. In the Parameters tab, add or modify the following information:
   - Hostname: and Port:

     The name (or IP address) and port number of the computer hosting the MySQL server. For
     example, localhost if the MySQL server is installed on the local computer. The default port
     value is 3306.
Connect Using MySQL Connections Manager

- **Username:**
  The name of a valid MySQL user account.

- **Password:**
  The password of the user account specified previously.

- **Default Schema:**
  A default schema name is required to open the connection. Select a name from the list.

6. Click **Test Connection** to verify the connection information.

7. Click **OK** to create and store the new connection. The new connection now appears in MySQL Connections Manager. Optionally, select the new connection from Connections Manager to add its tables, views, stored procedures, stored functions, and UDFs to the **Data Connections** node in Server Explorer.

### SSL Connections with Connections Manager

X Protocol connections can be configured to use SSL with PEM or PFX files. Connections must be created using the MySQL Connections Manager, which is supported by MySQL for Visual Studio 2.0.5 (or higher). MySQL Workbench provides similar support to add PEM files, but it does not support certificates in PFX format.

**Note**

X Plugin must be installed to support connections using X Protocol (see [Setting Up MySQL as a Document Store](#)).

In contrast, classic MySQL protocol connections support SSL PFX files only when you use MySQL Connections Manager to configure the connection.

To create a connection to a MySQL database using SSL encryption:

1. Click ![MySQL Connections Manager](image) in the Server Explorer menu bar to open the **MySQL Connections Manager** window.

2. Add and test a new basic connection (see [Basic Connections with MySQL Connections Manager](#)) or double-click an existing connection to modify it.

3. In the **SSL** tab, add a path to the SSL CA, SSL CERT, and SSL Key files within the SSL PEM area. Click **Test Connection** to verify the connection information. The next figure shows an example of SSL PEM values within this tab.

**Figure 7.12 MySQL Server Connection SSL Tab**
To configure SSL PFX (PKCS#12 format), select either the file-based or store-based option. Use the `.pfx` file extension to enable the correct certificate format.

4. Click **OK** to save the connection and return to the **MySQL Connections Manager** window.

   **Note**
   
   You must close and then reopen **MySQL Connections Manager** to apply the default schema.

5. Double-click the new SSL connection to add it to Server Explorer (or select the connection and click **OK**). To open the JavaScript or Python code editor, right-click the connection in Server Explorer and then select an editor.

### 7.5 Editing

Making edits in MySQL for Visual Studio.

After you have established a connection, for example, using the **Connect to MySQL** toolbar button, you can use auto-completion as you type or by pressing **Control + J**. Depending on the context, the auto-completion dialog can show the list of available tables, table columns, or stored procedures (with the signature of the routine as a tooltip). Typing some characters before pressing **Control + J** filters the choices to those items starting with that prefix.

#### 7.5.1 MySQL SQL Editor

The MySQL SQL Editor can be opened from the **MySQL toolbar** or by clicking **File, New, and File** from the Visual Studio main menu. This action displays the **New File** dialog.

**Figure 7.13 MySQL SQL Editor - New File**

![New File dialog]

From the **New File** dialog, select the MySQL template, select the **MySQL Script** document, and then click **Open**.

The MySQL SQL Editor will be displayed. You can now enter SQL code as required, or connect to a MySQL server. Click the **Connect to MySQL** button in the MySQL SQL Editor toolbar. You can enter the connection details into the **Connect to MySQL** dialog that is displayed. You can enter the server name, user ID, password and database to connect to, or click the **Advanced** button to select other connection string options. Click the **Connect** button to connect to the MySQL server. To execute your SQL code against the server, click the **Run SQL** button on the toolbar.
The results from queries are displayed in the **Results** tab and relevant information appears in the **MySQL Output** pane. The previous example displays the query results within a Result Grid. You can also select the Field Types, Execution Plan, and Query Stats for an executed query.

### 7.5.2 Code Editors

This section explains how to make use of the code editors in MySQL for Visual Studio.

#### Introduction

MySQL for Visual Studio provides access to MySQL objects and data without forcing developers to leave Visual Studio. Designed and developed as a Visual Studio package, MySQL for Visual Studio integrates directly into Server Explorer providing a seamless experience for setting up new connections and working with database objects.

The following MySQL for Visual Studio features are available as of version 2.0.5:

- JavaScript and Python code editors, where scripts in those languages can be executed to query data from a MySQL database.
- Better integration with the Server Explorer to open MySQL, JavaScript, and Python editors directly from a connected MySQL instance.
- A newer user interface for displaying query results, where different views are presented from result sets returned by a MySQL server like:
  - Multiple tabs for each result set returned by an executed query.
  - Results view, where the information can be seen in grid, tree, or text representation for JSON results.
Code Editors

- Field types view, where information about the columns of a result set is shown, such as names, data types, character sets, and more.
- Query statistics view, displaying information about the executed query such as execution times, processed rows, index and temporary tables usage, and more.
- Execution plan view, displaying an explanation of the query execution done internally by the MySQL server.

Getting Started

The minimum requirements are:

- MySQL for Visual Studio 2.0.5
- Visual Studio 2012
- MySQL 5.7.12 with X Plugin enabled (Code editors are not supported for use with MySQL 8.0 servers.)

To enable X Plugin for MySQL 5.7:

1. Open a command prompt and navigate to the folder with your MySQL binaries.
2. Invoke the mysql command-line client:
   ```
   mysql -u user -p
   ```
3. Execute the following statement:
   ```
   mysql> INSTALL PLUGIN mysqlx SONAME 'mysqlx.dll';
   ```

Important

The mysql.session user must exist before you can load X Plugin. mysql.session was added in MySQL 5.7.19. If your data dictionary was initialized using an earlier version you must run the mysql_upgrade procedure. If the upgrade is not run, X Plugin fails to start with the following error message:

There was an error when trying to access the server with user: mysql.session@localhost. Make sure the user is present in the server and that mysql_upgrade was ran after a server update.

Opening a Code Editor

Before opening a code editor that can run scripts against a MySQL server, a connection needs to be established:

1. Open the Server Explorer pane by clicking View.
2. Right-click the Data Connections node and select Add Connection.
3. In the Add Connection window, make sure the MySQL Data Provider is being used and fill in all the information.

Note

To enter the port number, click Advanced and set the Port among the list of connection properties.
4. Click Test Connection to ensure you have a valid connection, then click OK. The new connection with its tables, views, stored procedures, and functions now appears within the Data Connections list of Server Explorer.

5. Right-click the connection, select New MySQL Script, and then select the language of the editor (JavaScript or Python) to open a new MySQL script tab in Visual Studio.

To create a new editor for existing MySQL connections, you need only to do the last step.

**Using the Code Editor**

An open editor includes a toolbar with the actions that can be executed. The first two buttons in the toolbar represent a way to connect or disconnect from a MySQL server. If the editor was opened from the Server Explorer, the connection will be already established for the new script tab.

The third button is the Run button, the script contained in the editor window is executed by clicking it and results from the script execution are displayed in the lower area of the script tab.

**7.5.3 Editing Tables**

MySQL for Visual Studio contains a table editor, which enables the visual creation and modification of tables.

The Table Designer can be accessed through a mouse action on table-type node of Server Explorer. To create a new table, right-click the Tables node (under the connection node) and choose Create Table from the context-menu.

To modify an existing table, double-click the node of the table to modify, or right-click this node and choose the Design item from the context menu. Either of the commands opens the Table Designer.
Table Designer consists of the following parts:

- **Columns Editor** - a data grid on top of the Table Designer. Use the Columns grid for column creation, modification, and deletion. For additional information, see Section 7.5.3.1, “Column Editor”.

- **Indexes/Keys** window - a window opened from the Table Designer menu to manage indexes.

- **Relationships** window - a window opened from the Table Designer menu to manage foreign keys.

- **Column Properties** panel - a panel near the bottom of the Table Designer. Use the Column Properties panel to set advanced column options.

- **Properties** window - a standard Visual Studio Properties window, where the properties of the edited table are displayed. Use the Properties window to set the table properties. To open, right-click on a table and select the Properties context-menu item.

Each of these areas is discussed in more detail in subsequent sections.
To save changes you have made in the Table Designer, press either Save or Save All on the Visual Studio main toolbar, or press Control + S. If you have not already named the table, you will be prompted to do so.

**Figure 7.16 Choose Table Name**

After the table is created, you can view it in the Server Explorer.

**Figure 7.17 Newly Created Table**

The Table Designer main menu lets you set a primary key column, edit relationships such as foreign keys, and create indexes.

**Figure 7.18 Table Designer Main Menu**

### 7.5.3.1 Column Editor

You can use the Column Editor to set or change the name, data type, default value, and other properties of a table column. To set the focus to a needed cell of a grid, use the mouse click. Also you can move through the grid using Tab and Shift + Tab keys.

To set or change the name, data type, default value and comment of a column, activate the appropriate cell and type the desired value.

To set or unset flag-type column properties (NOT NULL, auto incremented, flags), select or deselect the corresponding check boxes. The set of column flags depends on the data type of the column.

To reorder columns, index columns or foreign key columns in the Column Editor, select the whole column to reorder by clicking the selector column on the left of the column grid. Then move the column by using Control+Up (to move the column up) or Control+Down (to move the column down) keys.
To delete a column, select it by clicking the selector column on the left of the column grid, then press the **Delete** button on a keyboard.

### 7.5.3.2 Column Properties

The **Column Properties** tab can be used to set column options. In addition to the general column properties presented in the Column Editor, in the **Column Properties** tab you can set additional properties such as Character Set, Collation and Precision.

![Column Properties Panel](image)

**Figure 7.19 Column Properties Panel**

### 7.5.3.3 Table Properties

To bring up Table Properties select the table and right-click to activate the context menu. Select **Properties**. The **Table Properties** dockable window will be displayed.
The following table properties are listed under table properties, and many are fully described in the SHOW TABLE STATUS MySQL documentation.

- **Auto Increment**: The next AUTO_INCREMENT value.
- **Average Row Length**: The AVG_ROW_LENGTH value.
- **Character Set**: The Charset value.
- **Collation**: The Collation value.
- **Comment**: Table comments.
- **Data Directory**: The directory used to store data files for this table.
- **Index Directory**: The directory used to store index files for this table.
- **Maximum Rows**: Value of the MAX_ROWS property.
- **Minimum Rows**: Value of the MIN_ROWS property.
- **Name**: Name of the table.
- **Row Format**: The ROW_FORMAT value.
- **Schema**: The schema this table belongs to.
- **Storage Engine**.
Note

In MySQL 5.5 and higher, the default storage engine for new tables is InnoDB. See Introduction to InnoDB for more information about the choice of storage engine, and considerations when converting existing tables to InnoDB.

The property **Schema** is read-only.

**Figure 7.21 Table Properties**

7.5.4 Editing Views

To create a new view, right-click the Views node under the connection node in Server Explorer. From the node's context menu, choose the **Create View** command. This command opens the SQL Editor.
You can then enter the SQL for your view, and then execute the statement.

To modify an existing view, double-click a node of the view to modify, or right-click this node and choose the Alter View command from a context menu. Either of the commands opens the SQL Editor.

All other view properties can be set in the Properties window. These properties are:

- **Catalog**: The TABLE_CATALOG.
- **Check Option**: Whether or not the WITH CHECK OPTION clause is present. For additional information, see The View WITH CHECK OPTION Clause.
- **Definer**: Creator of the object.
- **Definition**: Definition of the view.
- **Is Updatable**: Whether or not the view is Updatable. For additional information, see Updatable and Insertable Views.
- **Name**: The name of the view.
- **Schema**: The schema which owns the view.
- **Security Type**: The SQL SECURITY value. For additional information, see Stored Object Access Control.

Some of these properties can have arbitrary text values, others accept values from a predefined set. In the latter case, set the desired value with an embedded combobox.

The properties Is Updatable and Schema are read-only.

To save changes you have made, use either Save or Save All buttons of the Visual Studio main toolbar, or press Control + S.
7.5.5 Editing Indexes

Indexes management is performed using the **Indexes/Keys** dialog.

To add an index, select **Table Designer, Indexes/Keys...** from the main menu, and click **Add** to add a new index. You can then set the index name, index kind, index type, and a set of index columns.
7.5.6 Editing Foreign Keys

You manage foreign keys for InnoDB tables using the Foreign Key Relationships dialog.

To add a foreign key, select Table Designer, Relationships... from the main menu. This displays the Foreign Key Relationship dialog. Click Add. You can then set the foreign key name, referenced table name, foreign key columns, and actions upon update and delete.

To remove a foreign key, select it in the list box on the left, and click the Delete button.

To change foreign key settings, select the required foreign key in the list box on the left. The detailed information about the foreign key is displayed in the right hand panel. Change the desired values.
7.5.7 Editing Stored Procedures and Functions

To create a new stored procedure, right-click the Stored Procedures node under the connection node in Server Explorer. From the node's context menu, choose the Create Routine command. This command opens the SQL Editor.

To create a new stored function, right-click the Functions node under the connection node in Server Explorer. From the node's context menu, choose the Create Routine command.

To modify an existing stored routine (procedure or function), double-click the node of the routine to modify, or right-click this node and choose the Alter Routine command from the context menu. Either of the commands opens the SQL Editor.
Routine properties can be viewed in the **Properties** window. These properties are:

- Body
- Catalog
- Comment
- Creation Time
- Data Access
- Definer
- Definition
- External Name
- External Language
- Is Deterministic
- Last Modified
- Name
- Parameter Style
- Returns
- Schema
- Security Type
- Specific Name
- SQL Mode
- SQL Path
- Type

Some of these properties can have arbitrary text values, others accept values from a predefined set. In both cases, these values cannot be set from the properties panel.

You can also set all the options directly in the SQL Editor, using the standard **CREATE PROCEDURE** or **CREATE FUNCTION** statement.

To save changes you have made, use either **Save** or **Save All** buttons of the Visual Studio main toolbar, or press **Control + S**.
7.5.8 Editing Triggers

To create a new trigger, right-click the node of the table in which to add the trigger. From the node’s context menu, choose the **Create Trigger** command. This command opens the SQL Editor.

To modify an existing trigger, double-click the node of the trigger to modify, or right-click this node and choose the **Alter Trigger** command from the context menu. Either of the commands opens the SQL Editor.

To create or alter the trigger definition using SQL Editor, type the trigger statement in the SQL Editor using standard SQL.

**Note**

Enter only the trigger statement, that is, the part of the **CREATE TRIGGER** query that is placed after the **FOR EACH ROW** clause.

All other trigger properties are set in the Properties window. These properties are:

- **Definer**
- **Event Manipulation**
- **Name**
- **Timing**
Some of these properties can have arbitrary text values, others accept values from a predefined set. In the latter case, set the desired value using the embedded combo box.

The properties Event Table, Schema, and Server in the Properties window are read-only.

To save changes you have made, use either Save or Save All buttons of the Visual Studio main toolbar, or press Control + S. Before changes are saved, you will be asked to confirm the execution of the corresponding SQL query in a confirmation dialog.

To observe the runtime behavior of a stored routine and debug any problems, use the Stored Procedure Debugger. For additional information, see Section 7.10, “Debugging Stored Procedures and Functions”.

7.6 MySQL Project Items

This two-part tutorial uses MySQL MVC Item templates to set up a MVC web application. In the second part of the tutorial, a Windows Forms Item with MySQL connectivity is created.

Minimum Requirements

- MySQL 5.5 installed on a host that is accessible.
- MySQL for Visual Studio 1.2.5.
- Visual Studio 2012, the professional edition.
- MySQL Connector/NET is required to use web providers in the generated web application.

7.6.1 MySQL ASP.NET MVC Items

To add a MySQL MVC Item to an existing MVC project, first add a MySQL Entity Framework model. Skip this step if you have already done this.

Configure the project to use MySQL with an Entity Framework. There are two ways to do this:

- Manually add the references needed (EntityFramework, MySql.Data &MySql.Data.Entity), and add the required configuration to the web.config configuration file
- Or (preferred), take advantage of the MySQL Website Configuration tool, which allows either Entity Framework 5 or 6 with MySQL. For additional information about this tool, see Section 7.7, “MySQL Application Configuration Tool”.

Once you have configured the project to use MySQL with Entity Framework, proceed to create the model using the standard ADO.NET Entity Data Model wizard. For MySQL MVC Item Templates, you need to add the model under the “Models” folder, as illustrated below:
Figure 7.28 ADO.NET Entity Data Model

Contains an entity model from a database. Object-layer code is generated from the model. This option also lets you specify the database connection, settings for the model, and database objects to include in the model.
Figure 7.29 Choose or create a new MySQL connection

Figure 7.30 Creating a new MySQL connection
After selecting the MySQL connection, you need to select the database objects to include in the model.  

**Important**  
The **Pluralize or singularize generated object names** option must remain unchecked, otherwise the MySQL MVC Item Template will not function properly.

*Figure 7.31 Selecting the database object to include in the model*

Click **Finish** to generate the model, as demonstrated below:
Now, generate a new MySQL MVC Item. Right-click on the project, and select Add New Item from the contextual menu.

This launches the Add New Item wizard. The MySQL menu offers two options: MySQL New MVC Item and MySQL New Windows Form. Select MySQL New MVC Item, and then click Add.
This opens the **MVC Item Template** dialog. Now select the MySQL model and entity that you want to use to create the MVC item. The model dropdown list is populated based on all the MySQL Entity Framework models available in the project, entities dropdown list is populated with entities available for the selected model.
After selecting the model and entity to create the item, click **Finish**, and a new controller and view matching the selected entity will be added to the project. These contain the necessary back end code to render the entity data.
You can now execute the application. In our example we used the Sakila database and generated an Actor controller:

**Figure 7.38 The Actor View**

<table>
<thead>
<tr>
<th>actor_id</th>
<th>first_name</th>
<th>last_name</th>
<th>last_update</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PENELLOPE</td>
<td>GUINESS</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>2</td>
<td>NICK</td>
<td>WAHLBERG</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>3</td>
<td>ED</td>
<td>CHASE</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>4</td>
<td>JENNIFER</td>
<td>DAVIS</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>5</td>
<td>JOHNNY</td>
<td>LOLLOBIRIGIDA</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>6</td>
<td>BETTE</td>
<td>NICHOLSON</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>7</td>
<td>GRACE</td>
<td>MOSTEL</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>8</td>
<td>MATTHEW</td>
<td>JOHANSSON</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>9</td>
<td>JOE</td>
<td>SWANK</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
<tr>
<td>10</td>
<td>CHRISTIAN</td>
<td>GABLE</td>
<td>2/15/2006 4:34:33 AM</td>
</tr>
</tbody>
</table>

© 2015 - My ASP.NET Application

### 7.6.2 MySQL Windows Forms Items

This portion of the tutorial describes how to create a Windows Form with MySQL connectivity. The item template to use for this operation is named **MySQL New Windows Form**. To open the Add New Item window, right-click your windows form application and select **MySQL** from the list of installed items.

The item template for adding a new MySQL Windows Form is similar to the MySQL MVC item template (see Section 7.6.1, “MySQL ASP.NET MVC Items”), but with three major differences:

- You can create the MySQL Entity Framework model under the root path of the project.

**Figure 7.39 A MySQL Entity Framework model created in a Windows Form Application**

- When selecting the desired entity, you can also select the layout type in which the new form will display the entity data.
Figure 7.40 The “MySQL Windows Form” Item Template dialog, with the layout options

- A Resources folder is added to the project that contains images used by the icons for the generated form.

Figure 7.41 The Resources Folder and New Form

The new form will have all the necessary back-end code to display the entity data, with the user interface (UI) based on the previously selected layout.
7.7 MySQL Application Configuration Tool

The MySQL Application Configuration tool (previously named MySQL Website Configuration) enables you to configure options for the following categories with MySQL as the database provider: entity...
framework, membership (advanced or simple), roles, profiles, session state, site map, and web personalization. With this MySQL for Visual Studio feature, you can configure multiple provider pages in sequence and the tool modifies your configuration files accordingly.

Editing configuration files manually can be problematic. The MySQL Application Configuration tool simplifies the task by providing the relevant options for each web (or application) project in a graphical, wizard-like format that permits you to navigate among the provider categories. The tool then adds, modifies, or removes entries from the App.config file, which applies to settings for non-web projects only, and the Web.config file for your web-based projects.

The MySQL Application Configuration tool appears as a small icon on the Solution Explorer toolbar in Visual Studio, as shown in the following figure. The icon is visible only when a project is active (with a connection to MySQL), and Connector.NET is installed. Clicking the MySQL Application Configuration icon launches the tool and displays the options for setting up Entity Framework support.

Figure 7.44 MySQL Application Configuration Tool

The remainder of this chapter describes each item that you can configure using the MySQL Application Configuration tool.

7.7.1 Entity Framework

The MySQL Application Configuration tool downloads the latest entity framework and MySQL Connector.NET assemblies from the NuGet gallery to keep the assembly versions synchronized. After the configuration is applied, the tool adds the following references to your project:

- EntityFramework
- EntityFramework.SqlServer
- MySql.Data
- MySql.Data.EntityFramework

Unlike the other web providers supported by this tool, the entity framework options can be applied to either non-web or web applications. The context of your project determines which configuration file the tool updates. The next sections describe the details for setting options with non-web and web projects.

Note

MySQL for Visual Studio does not support Entity Framework Core. For the current support profile, see Minimum Requirements.
**Setting Options for Non-Web Applications**

The MySQL Application Configuration tool synchronizes the latest version of MySQL Connector/.NET and Entity Framework 6 assemblies and then adds the required references to your application's App.config file. When started in the context of a non-web application, the tool lists entity framework as the only item and MySQL for Visual Studio 1.2.9 (or higher) is a prerequisite. For an example of when you might use the tool, see Tutorial: Using an Entity Framework Entity as a Windows Forms Data Source.

To enable the tool, select **Use MySQL with Entity Framework**, ensure that **Entity Framework 6.x** is selected, and then click **Finish** (see the figure that follows).

**Figure 7.45 MySQL Application Configuration Tool for Entity Framework (non-web)**

**Setting Options for Web Applications**

In the context of a web application, entity framework is the first page of options that you can enable. After enabling (or skipping) the entity framework option, the wizard-like tool steps through several additional web provider pages (see Section 7.7.2, "Web Providers"). You can enable entity framework alone and skip the other web providers, or selectively enable other providers in the same session.

With MySQL for Visual Studio 1.2.9 (or higher) installed, Entity Framework 6 is the only version to select, as the following figure shows. Previous versions of MySQL for Visual Studio permit you to configure your application to use Entity Framework 5 or 6. (Prior to version 1.2.9, this tool was named MySQL Website Configuration.) The current version of MySQL Connector/.NET no longer supports Entity Framework 5.

To include entity framework options, select **Use MySQL with Entity Framework**, ensure that an **Entity Framework** version is selected, and then click **Next** to advance to the membership provider options. To skip configuring entity framework options, deselect the **Use MySQL with Entity Framework** check box.
For information about the options on the membership (or simple membership), roles, profiles, session state, site map, and web personalization pages, see Section 7.7.2, “Web Providers”.

### 7.7.2 Web Providers

The MySQL Application Configuration tool establishes MySQL as the database provider for one or more web providers by making changes to your application’s `Web.config` file. You can choose only one of the two membership providers: membership or simple membership. All of the other web providers may be enabled independently or together. The configuration entries for each web provider managed by the MySQL Application Configuration tool are included in the sections that follow.

- Membership Provider
- Simple Membership Provider
- Roles Provider
- Profiles Provider
- Session State Provider
- Site Map Provider
- Web Personalization Provider

Web-provider pages share several common elements, as the membership provider page shows.
Check box to enable the provider. Web providers are enabled when you select the check box on the page. When selected, the page enforces requirement-checking specific to the provider. For example, if you enable any provider, the configuration file for your application must include a connection string. If the connection string is missing, the tool returns a warning when you attempt to move to the next provider. To skip (or disable) an individual web provider, deselect the check box.

Name. Each enabled web provider requires a value to specify the application name. If you do not provide a value, then the default value creates an application name for you. The value is associated with the `applicationName` property in the `Web.config` file.

Connection String. The `Web.config` file stores a single connection string for all of the MySQL web-providers. Each web-provider page includes an area for entering a connection string, however only the last entry is saved. For example, if you set it in the first web provider and also set it on the third web provider, the connection string from the third provider page is saved. You can enter a connection string directly in the text box provided or click **Edit** to use a visual editor to help you sort and select the various connection string options (see Section 7.7.3, “Using the MySQL Connection String Editor”).

Autogenerate Schema. Select the **Autogenerate Schema** option check box to ensure that the necessary schemas are created automatically for the web provider being configured. These schemas are used to store MySQL web provider information. The database used for storage is the one specified in the connection string.

Write exceptions to event log. Select the **Write exceptions to event log** option check box to ensure that exceptions generated by the application are written to the Windows event log.

Membership Provider

In addition to the standard membership provider, there is also a simple membership provider. You can only choose one of these two membership providers. To use the membership provider, select **Use MySQL to manage my membership records** to enable the page. You can now enter the name of the application that you are creating the configuration for. You can also enter a description for the application. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up a membership provider, a new section is added to the web configuration file.

```xml
<membership defaultProvider="MySQLMembershipProvider">
    <providers>
        <remove name="MySQLMembershipProvider" />
    </providers>
</membership>
```
With one of the membership providers configured, click Next to advance to the roles provider page.

**Simple Membership Provider**

The simple membership provider options are similar to those of the membership provider, but with fewer properties in the configuration file. To enable, check the **Use MySQL to manage my simple membership records**.

**Note**

The simple membership provider is not supported by MySQL Connector/NET 8.0 and cannot be enabled if you have the 8.0 version of the connector installed.

The MySQL simple membership provider handles the website membership tasks with ASP.NET. This provider is a simpler version of the ASP.NET Membership provider, and it can also work with OAuth Authentication. For additional information about using OAuth authentication, see Adding OAuth Authentication to a Project.

The required configuration options for the simple membership provider are: a name for the connection string and a connection string that contains a valid database with a local or remote MySQL server instance, a user table to store the credentials, and column names for the **User ID** and **User Name** columns.

Select the **Auto Create Tables** option to create the required tables when adding the first user to the table. After setting up a membership provider, a new section is added to the web configuration file.

After setting up one of the membership providers, click Next to configure the roles provider page.

**Roles Provider**

Click **Use MySQL to manage my roles** to enable this provider page. The page includes the following options to edit: the connection string, the application name, and a description of the provider. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up a roles provider, a new section is added to the web configuration file.
Web Providers

Click **Next** to configure the profiles provider page.

**Profiles Provider**

Click **Use MySQL to manage my profiles** to enable this provider page. The page includes the following options to edit: the connection string, the application name, and a description of the provider. The **Autogenerate Schema**, **Write exceptions to event log**, and **Callback for session end event** options can be selected for this web provider.

After setting up a profiles provider, a new section is added to the web configuration file.

Click **Next** to configure the session state provider page.

**Session State Provider**

Click **Use MySQL to manage my ASP.Net session state** to enable this provider page. The page includes the following options to edit: the connection string, the application name, and a description of the provider. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up a session provider, a new section is added to the web configuration file.

Click **Next** to configure the site map provider page.

**Site Map Provider**

The site map provider builds a site map from a MySQL database, and builds a complete tree of the SitemapNode objects. It also provides methods so that the generated nodes can be read from the site map. Click **Use MySQL to manage my ASP.NET site map** to enable this provider page.

The required configuration options: A name for the application and a connection string that contains a valid database with a local or remote MySQL server instance. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up the site map provider, a new section is added to the web configuration file.
Click **Next** to proceed to the web personalization configuration page:

**Web Personalization Provider**

The web personalization provider is used when a website application needs to store persistent information for the content and layout of the Web Parts pages that are generated by a Web Parts personalization service. This provider should be used along with the membership, roles, and profiles providers. Click **Use MySQL to manage my ASP.NET personalization provider** to enable this provider page.

The required configuration options: A name for the application and a connection string that contains a valid database with a local or remote MySQL server instance. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up the web personalization provider, a new section is added to the web configuration file.

When you have selected the web personalization options, click **Finish** to write the changes for all web providers to the **Web.config** file and close the tool.

**7.7.3 Using the MySQL Connection String Editor**

The MySQL Connection String Editor lists and describes all supported connection-string options for the version of MySQL Connector/.NET that you have installed. To open the editor from any web-provider page (see the **Connection String** entry box), click **Edit**.

By default, the editor lists connection-string options by category (see the figure that follows). Alternatively, you can sort the list alphabetically using the sorting buttons provided. The main list consists of options on the left and values on the right of a split area. To add a text value, such as the database name, select the option in the list (**Database**, in this example), and then type the name of the database (**sakila**) after the cursor. Some connection-string options have predefined values and show a down arrow when selected.

When you select a connection-string option, a brief description is shown in an information box below the list. After you add an option value and then select another option, the editor appends the new option/value pair to the existing connection string. Click **OK** to return to the web-provider page, which now includes the modified connection string.
Figure 7.48 MySQL Application Configuration Tool - Connection String Editor

Note
Defined connection strings are automatically loaded and available in editor, whether they were created manually in `<web.config>` for each application or previously using this tool.

7.8 MySQL Data Export Tool

MySQL for Visual Studio has a data export tool that creates a dump file for a MySQL database.
Creating a Dump of an existing MySQL Database

In order to open a new window for the MySQL Data Export tool, the user must create a new connection using the Server Explorer Window inside Visual Studio. Once the connection is established, a contextual menu can be opened by right clicking on the connection node. From the menu, choose the MySQL Data Export option. A new tabbed-window opens for the current connection. The user can select one or more databases to include in the dump.
Follow these steps to create a dump for the MySQL Databases:

1. Select all the databases and their objects to be included in the dump.
2. It is very important to select the desired settings for the dump: whether the dump will include the data, whether the insert operations will be logged in extended mode, and so on. In the main window of the MySQL Database Export tool are shown the basic options for the dump. Also, by clicking on the Advanced button, more specific options can also be selected.

Figure 7.52 MySQL for Visual Studio Data Export Tool: Advanced Options

3. When the selection of the options is done, give a name to the result file that will be created. If no path is given for the result file, the default path to be used is My Documents under the user's folder.

Figure 7.53 MySQL for Visual Studio Data Export Tool: Generating the Dump File
4. A filter can be applied on the list of schemas for the selected connection. With it, the user can easily locate the databases to be included in the dump.

**Figure 7.54 MySQL for Visual Studio Data Export Tool: Filtering the Schemas**

5. After selecting the options and the name for the dump file, the user can click the **Export** button, which generates the dump.

**Figure 7.55 MySQL for Visual Studio Data Export Tool: Viewing the Generated Script**
Creating a Dump of an existing MySQL Database

Each dump can have different settings. After configuring the dump operation, the settings can be saved into a setting file for later use. This file includes: the connection selected, the name of the file for the dump, and the database or databases and the objects selected for dumping. The file extension for the setting file is `.dumps`.

Figure 7.56 MySQL for Visual Studio Data Export Tool: Saving a Setting File

A saved setting file can be loaded into the MySQL Data Export tool by clicking the Load Settings button.

Figure 7.57 MySQL for Visual Studio Data Export Tool: Opening a Setting File
7.9 DDL T4 Template Macro

Convert an Entity Framework model to MySQL DDL code. Starting with a blank model, you can develop an entity model in Visual Studio's designer. Once the model is created, you can select the model's properties, and in the Database Script Generation category of the model's properties, the property **DDL Generation** can be found. Select the value **SSDLToMySQL.tt(VS)** from the drop-down list.

**Figure 7.58 DDL T4 Template Macro - Model Properties**

Right-clicking the model design area displays a context-sensitive menu. Selecting **Generate Database from Model** from the menu displays the **Generate Database Wizard**. The wizard can then be used to generate MySQL DDL code.
7.10 Debugging Stored Procedures and Functions

The stored procedure debugger provides facilities for setting breakpoints, stepping into individual statements (Step Into, Step Out, Step Over), evaluating and changing local variable values, evaluating breakpoints, and other debugging tasks.

Privileges

The debugger recreates at the start of each debug session a serversidedebugger database in your server. This database helps to track the instrumented code and implement observability logic in the debugged routine. Your current connection needs to have privileges to create that database, and its associated stored routines, functions, and tables.

The debugger makes changes behind the scenes to temporarily add instrumentation code to the stored routines that you debug. You must have the ALTER ROUTINE privilege for each stored procedure, function, or trigger that you debug. (Including procedures and functions that are called, and triggers that are fired, by a procedure that you are debugging.)

Starting the Debugger

To start the debugger, follow these steps:

2. Expand the Stored Procedures folder. Only stored procedures can be debugged directly. To debug a user-defined function, create a stored procedure that calls the function.
3. Click on a stored procedure node, then right-click and from the context menu choose Debug Routine.
At this point, Visual Studio switches to debug mode, opening the source code of the routine being debugged in step mode, positioned on the first statement.

If the initial routine you debug has one or more arguments, a pop-up will show up with a grid (a row per each argument and three columns: one for the argument, one for the argument value (this is editable) and one for nullifying that argument value (a checkbox). After setting up all the argument values, you can press OK to start the debug session, or Cancel to cancel the debug session.
How the Debugger Functions

To have visibility into the internal workings of a stored routine, the debugger prepares a special version of the procedure, function, or trigger being debugged, instrumented with extra code to keep track of the current line being stepped into and the values of all the local variables. Any other stored procedures, functions, or triggers called from the routine being debugged are instrumented the same way. The debug versions of the routines are prepared for you automatically, and when the debug session ends (by either pressing F5 or Shift + F5), the original versions of the routines are automatically restored.

A copy of the original version of each instrumented routine (the version without instrumentation) is stored in the AppData\Roaming\MySqlDebuggerCache folder for the current Windows user (the path returned by calling System.Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) in .NET, plus appending MySqlDebuggerCache. There is one file for each instrumented routine, named routine_name.mysql. For example, in Windows 7, for a user named fergs, the path is C: \Users\fergs\AppData\Roaming\MySqlDebuggerCache.

Two threads are used, one for the debugger and one for the routine being debugged. The threads run in strict alternation, switching between the debugger and the routine as each statement is executed in the stored routine.

Basic Debugging Operations

The debugger has the same look and feel as the standard Visual Studio debuggers for C#, VB.NET or C++. In particular, the following are true:

Locals and Watches

- To show the Locals tab, choose the menu item Debug, Windows, Locals.
The **Locals** tab lists all the variables available in the current scope: variables defined with `DECLARE` at any point in the routine, argument parameters, and session variables that are referenced.

- If the last step operation changes the value of a local, its value will be highlighted in red (until another statement is executed or stepped).
- You can change the value of any local.
- To show the **Watch** tab, choose the menu item **Debug, Windows, Watch**.
  
  To define a watch, type any valid MySQL expression, optionally including function calls. If the watch evaluation makes sense in the current context (current stack frame), it will show its value, otherwise it will show an error message in the same row the watch was defined.
  
  - When debugging a trigger, in addition to any locals declared or session variables referenced, the new and old object (when applicable) will be listed. For example in a trigger for `INSERT`, for a table defined like:
    ```
    create table t1( id int, myname varchar(50));
    ```
    the locals will list the extra variables `new.id` and `new.myname`. For an `UPDATE` trigger, you will also get the extra variables `old.id` and `old.myname`. These variables from the new and old objects can be manipulated the same way as any ordinary local variable.

**Figure 7.63 Debugging a Trigger**

```
create trigger tTriggerTable BEFORE INSERT on test6.triggerTable for each row
set new.myid = new.myid + 1;
end
```

**Call Stack**

- To show the **Call Stack** tab, choose the menu item **Debug, Windows, Call Stack**.
  
  The stack trace (in the **Call Stack** tab) will list all the stack traces, one for each routine invocation. The one with a yellow mark is the current stepping point. Clicking in another will activate in the editor the tab for that routine source, highlighting in green the last statement stepped.
Stepping

- Stepping of a new routine starts in the first executable instruction (excluding declares, handlers, cursor declarations, and so on).
Figure 7.65 Debug Stepping

```sql
CREATE PROCEDURE `spTest`()
BEGIN
    declare n int;
    set n = 1;
    while n < 10 do
        begin
            set n = n + 1;
        end;
    end while;
end
```

Figure 7.66 Function Stepping (1 of 2)

```sql
CREATE PROCEDURE `SimpleNonScalar`
BEGIN
    update CalcData set z = BeSum( x, y);
end
```
• To step into the code of a condition handler, the condition must be triggered in the rest of the MySQL routine.

• The next statement to be executed is highlighted in yellow.

• To continue stepping, you can choose between **Step Into** (by pressing F11), **Step Out** (by pressing F10) or **Step Over** (by pressing Shift + F11).

• You can step out of any of functions, triggers or stored procedures. If you step from the main routine, it will run that routine to completion and finish the debug session.

• You can step over stored procedure calls, stored functions, and triggers. (To step over a trigger, step over the statement that would cause the trigger to fire.)

• When stepping into a single statement, the debugger will step into each individual function invoked by that statement and each trigger fired by that statement. The order in which they are debugged is the same order in which the MySQL server executes them.

• You can step into triggers triggered from **INSERT**, **DELETE**, **UPDATE**, and **REPLACE** statements.

• Also, the number of times you enter into a stored function or trigger depends on how many rows are evaluated by the function or affected by the trigger. For example, if you press F11 (**Step Into**) into an **UPDATE** statement that modifies three rows (calling a function for a column in the **SET** clause, thus invoking the function for each of the three rows), you will step into that function three times in succession, once for each of the rows. You can accelerate this debug session by disabling any breakpoints defined in the given stored function and pressing **Step Over** to step out. In this example, the order in which the different instances of the stored function are debugged is server-specific: the same order used by the current MySQL server instance to evaluate the three function invocations.

**Breakpoints**
• To show the **Breakpoints** tab, choose the menu item **Debug, Windows, Breakpoints**.

• The **Breakpoints** tab will show all the breakpoints defined. From here, you can enable and disable breakpoints one by one or all at once (using the toolbar on top of the **Breakpoints** tab).

• You can define new breakpoints only in the middle of a debug session. Click in the left gray border of any MySQL editor, or click anywhere in a MySQL editor and press **F9**. In the familiar Visual Studio way, you press **F9** once to create a breakpoint in that line, and press it again to remove that breakpoint.

• Once a breakpoint is defined, it will appear enabled (as filled red circle left to the current row if that line is a valid statement to put a breakpoint) or disabled (as a non-filled red circle left to the current row if that row is not valid to put a breakpoint).

• To define conditional breakpoints, after creating the breakpoint, right click in the red dot and choose **Condition**... There you can put any valid MySQL expression and state if the condition is **Is True** or **Has changed**. The former will trigger the breakpoint every time the condition is true, the latter every
time the condition value has changed. (If you define a conditional breakpoint, it is not enough to step into the line with the breakpoint defined to trigger such a breakpoint.)

Figure 7.68 Conditional Breakpoints

![Conditional Breakpoints Diagram]
Figure 7.69 Expressions and Breakpoints

- To define pass count breakpoints, after creating the breakpoint, right click in the red dot and choose Hit Count... In the pop-up dialog, define the specific condition to set. For example, break when the hit count is equal to and a value 3 will trigger the breakpoint the third time it is hit.

Other Features

- To abort the debug session (and the execution of the current call stack of routines), press Shift + F5.
- To run the routine to completion (or until next breakpoint hit), press F5.
- For all functionality you can use (in addition to the shortcuts documented), see the options in the Debug menu of Visual Studio.

Limitations

- Code being debugged must not use get_lock or release_lock MySQL functions, since they are used internally by the debugger infrastructure to synchronize the debugger and the debugged routine.
- Code being debugged must avoid using any transaction code (START TRANSACTION, COMMIT, ROLLBACK) due to the possibility of wiping out the contents of the debugger tables. (This limitation may be removed in the future).
- You cannot debug the routines in the serversidedebugger database.
- The MySQL server running the routine being debugged can be any MySQL server version after 5.0, and running on any supported platform.
- Always run debug sessions on test and development servers, rather than against a MySQL production server, because debugging can cause temporary performance issues or even deadlocks. The instrumented versions of the routines being debugged use locks that might not pertain to the rest of the production code.
Keyboard Shortcuts

The following list summarizes the keyboard shortcuts for debugging:

- **F9**: Toggles breakpoints
- **F11**: Step into once
- **F10**: Step over once
- **Shift + F11**: Step out once
- **F5**: Run
- **Shift + F5**: Abort current debug session

### 7.11 MySQL for Visual Studio Frequently Asked Questions

#### Questions

- **7.11.1**: How do I know if MySQL for Visual Studio is installed?

#### Questions and Answers

**7.11.1**: How do I know if MySQL for Visual Studio is installed?

Open Visual Studio and go to **View, Toolbars**, and look for (and enable) the **MySQL** toolbar. Or, open **MySQL Installer** and look for the MySQL for Visual Studio product.