



MySQL connection strings



MySQL Connector/Net

Standard

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

Specifying TCP port

```
Server=myServerAddress; Port=1234; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

The port 3306 is the default MySQL port.

The value is ignored if Unix socket is used.

Multiple servers

Use this to connect to a server in a replicated server configuration without concern on which server to use.

```
Server=serverAddress1, serverAddress2,  
serverAddress3; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

Using encryption (old)

This one activates SSL encryption for all data sent between the client and server. The server must have a certificate installed.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Encrypt=true
```

This option is available for Connector/NET version 5.0.3 through 6.2.1. From 6.2.1 use the SslMode option instead.

Using encryption (new)

Use SSL if the server supports it, but allow connection in all cases

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; SslMode=Preferred
```

This option is available from Connector/NET version 6.2.1

Force encryption

Always use SSL. Deny connection if server does not support SSL.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; SslMode=Required
```

This option is available from Connector/NET version 6.2.1

SSL with a file-based certificate

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; SSL Mode=Re
```

This option is available from Connector/NET version 6.2.1

SSL with a personal store-based certificate

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; SSL Mode=Re
```

This option is available from Connector/NET version 6.2.1

SSL with a thumbprint specific personal store-based certificate

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; SSL Mode=Re
```

This option is available from Connector/NET version 6.2.1

Disallow batches

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; AllowBatch=
```

Allow User Variables

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; AllowUserVar
```

This option is available from Connector/NET version 5.2.2

Invalid DateTime's 1

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; AllowZeroDat
```

Returns a MySqlDateTime object for invalid values and a System.DateTime object for valid values.

Invalid DateTime's 2

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; ConvertZeroL
```

Returns System.DateTime.MinValue valued System.DateTime object for invalid values and a System.DateTime object for valid values.

Disable transaction participation

The use of auto enlist transactionscope (default behaviour) could cause trouble in medium trust environments.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; AutoEnlist=
```

Skip parameter checks for stored routines

Default behaviour is that parameters for stored routines (stored procedures) are checked against the server

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; CheckParameter
```

Some permissions and value casting related errors reported fixed when using this connection option.

Skip parameter type and order matching for stored procedures

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; UseProcedure
```

The default behaviour is to read tables `mysql.proc/INFORMATION_SCHEMA.ROUTINES` and try to map provided command parameter values to the called procedures parameters and type cast values accordingly.

This can be troublesome if permissions to the (aforementioned) sproc info tables are insufficient.

The driver will not automatically map the parameters so you must manually set parameter types and you must also make sure to add the parameters to the command object in the exact order as appeared in the procedure definition.

This option is available from Connector/NET version 5.0.4

Use Table Caching

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; tablecache=
```

Specifying `DefaultTableCacheAge` is optional, default value is 60 seconds.

This option is available from Connector/NET version 6.4

Count changed rows instead of found rows

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; UseAffectedF
```

This option is available from Connector/NET version 5.2.6

Compress network communication between client and server

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; UseCompressi
```

Log inefficient database operations

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; UseUsageAdvi
```

Enable performance counters

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; UsePerformar
```

Connection Pooling

From version 6.2 idle connections are removed from the pool, freeing resources on the client (sockets) and the server (sockets and threads). Do not manually keep (global) connections and open close. Keep connection object creation and disposal as tight as possible, this might be counterintuitive but pooling mechanisms will take care of caching well and your code will be cleaner.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Pooling=True
```

This is the default behaviour.

Connection Pool size

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; MinimumPoolSize=0; MaximumPoolSize=100
```

Default values are 0 and 100.

Disable connection pooling

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Pooling=False
```

Connection state reset when obtained from pool

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; ConnectionResetWhenObtainedFromPool=True
```

Makes an additional round trip to the server when obtaining a connection from the pool and connection state will be reset.

Recycle connections in pool

This is useful in load balancing scenarios when available servers change you don't want 100 constant connections in the pool pointing to only one server.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; ConnectionLifetime=30
```

Specified in seconds, the amount of time after connection object creation the connection will be destroyed. Destruction will only happen when connections are returned to pool.

Do not update server settings on connections in pool

A connection might be long lived in the pool, however the connections server settings are updated (SHOW VARIABLES command) each time returned to the pool. This makes the client use of the connection object up to date with the correct server settings. However this causes a round trip and to optimize pooling performance this behaviour can be turned off.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; CacheServerSettings=True
```

This option is available from Connector/NET version 6.3

Use Windows authentication

```
Server=myServerAddress; Database=myDataBase; IntegratedSecurity=yes; Uid=auth_windows;
```

This option is available from Connector/NET version 6.4.4

The Windows Native Authentication Plugin must be installed for this to work.

Keep TCP Sockets alive

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Keepalive=1
```

Number of seconds between each keep-alive package send.

Use BINARY(16) GUIDs

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; OldGuids=Tr
```

This option is available from Connector/NET version 6.1.1

Disable Stored procedure cache

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; ProcedureCac
```

The default is 25, meaning that stored procedure meta data (such as input/output data types etc) for the latest 25 called procedures will be cached in client memory.

This option is available from Connector/NET version 5.0.2

Allow square brackets around symbols (instead of backticks)

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; sqlservermoc
```

This enables Visual Studio wizards that bracket symbols with [] to work with Connector/Net. This option incurs a performance hit, so should only be used if necessary.

This option is available from Connector/NET version 6.3.1

Specifying default command timeout

Use this one to specify a default command timeout for the connection. Please note that the property in the connection string does not supercede the individual command timeout property on an individual command object.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; default comm
```

This option is available from Connector/NET version 5.1.4.

Specifying connection attempt timeout

Use this one to specify the length in seconds to wait for a server connection before terminating the attempt and receive an error.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Connection 1
```

Inactivating prepared statements

Use this one to instruct the provider to ignore any command prepare statements and prevent corruption issues with server side prepared statements.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Ignore Prepared Statements
```

The option was added in Connector/NET version 5.0.3 and Connector/NET version 1.0.9.

Specifying network protocol

Use this one to specify which network protocol to use for the connection.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Protocol=socket
```

"socket" is the default value used if the key isn't specified. Value "tcp" is an equivalent for "socket".

Use "pipe" to use a named pipes connection, "unix" for a Unix socket connection and "memory" to use MySQL shared memory.

Shared memory protocol

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Protocol=memory
```

It's possible to explicit set the shared memory object name used for communication.

Named pipes protocol

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Protocol=pipe
```

It's possible to explicit set the pipe name used for communication, if not set, 'mysql' is the default value.

Named pipes alternative

```
Server=myServerAddress; Port=-1; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

It is the port value of -1 that tells the driver to use named pipes network protocol. This is available on Windows only. The value is ignored if Unix socket is used.

Unix socket connection

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; Protocol=unix
```

It's possible to explicit set the shared memory object name used for communication.

Specifying character set

Use this one to specify which character set to use to encode queries sent to the server.

```
Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; CharSet=utf8
```

Note! Use lower case value utf8 and not upper case UTF8 as this will fail.

Note that resultsets still are returned in the character set of the data returned.

MySQLDriverCS

Standard

```
Location=myServerAddress; Data Source=myDataBase; User ID=myUsername; Password=myPassword;
```

SevenObjects MySqlConnection

Standard

```
Host=myServerAddress; UserName=myUsername; Password=myPassword; Database=myDataBase;
```

dotConnect for MySQL (former MyDirect.NET and Core Labs MySQLDirect.NET)

Standard

```
User ID=root; Password=myPassword; Host=localhost; Port=3306; Database=myDataBase; Direct
```

MySQL OLEDB

Standard

```
Provider=MySQLProv; Data Source=mydb; User Id=myUsername; Password=myPassword;
```

eInfoDesigns.dbProvider


Standard

```
Data Source=myServerAddress; Database=myDataBase; User ID=myUsername; Password=myPassword;
```

.NET Framework Data Provider for OLE DB

Use an OLE DB provider from .NET

```
Provider=any oledb provider's name; OleDbKey1=someValue; OleDbKey2=someValue;
```

See the respective OLEDB provider's connection strings options. The .net OleDbConnection will just pass on the connection string to the specified OLEDB provider. Read more [here](#) .

MySQL Connector/ODBC 5.2

Unicode version of the driver

```
Driver={MySQL ODBC 5.2 UNICODE  
Driver}; Server=localhost; Database=myDataBase; User=myUsername; Password=myPassword; Opt
```

"MySQL ODBC 5.2 **UNICODE** Driver" is the new name for "MySQL ODBC **5.1w** Driver".

"MySQL ODBC 5.2 **ANSI** Driver" is the new name for "MySQL ODBC **5.1a** Driver".

Local database

```
Driver={MySQL ODBC 5.2 ANSI  
Driver}; Server=localhost; Database=myDataBase; User=myUsername; Password=myPassword; Opt
```

On 64 bit machine

It seems we need to point on MSDASQL for the (32 bit) ODBC driver to work on 64 bit machines.

```
Provider=MSDASQL; Driver={MySQL ODBC 5.2 ANSI  
Driver}; Server=localhost; Database=myDataBase; User=myUsername; Password=myPassword; Opt
```

Remote database

```
Driver={MySQL ODBC 5.2 ANSI  
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo
```

Specifying TCP/IP port

```
Driver={MySQL ODBC 5.2 ANSI  
Driver}; Server=myServerAddress; Port=3306; Database=myDataBase; User=myUsername; Passwor
```

The driver defaults to port value 3306, if not specified in the connection string, as 3306 is the default port for MySQL.

Specifying character set

```
Driver={MySQL ODBC 5.2 ANSI  
Driver}; Server=myServerAddress; charset=UTF8; Database=myDataBase; User=myUsername; Pass
```

Specifying socket

This one specifies the Unix socket file or Windows named pipe to connect to. Used only for local client connections.

```
Driver={MySQL ODBC 5.2 ANSI  
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo
```

On Windows, the socket variable is the name of the named pipe that is used for local client connections. The default value is MySQL.

On Unix platforms, the socket variable is the name of the socket file that is used for local client connections. The default is /tmp/mysql.sock.

Using SSL

```
Driver={MySQL ODBC 5.2 ANSI  
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo  
cert.pem; sslkey=c:\client-key.pem; sslverify=1;  
Option=3;
```

SSLCA specifies the path to a file with a list of trust SSL CAs

SSLCERT specifies the name of the SSL certificate file to use for establishing a secure connection.

SSLKEY specifies the name of the SSL key file to use for establishing a secure connection.

MySQL Connector/ODBC 5.1

Local database

```
Driver={MySQL ODBC 5.1  
Driver}; Server=localhost; Database=myDataBase; User=myUsername; Password=myPassword; Opt
```

On 64 bit machine

It seems we need to point on MSDASQL for the (32 bit) ODBC driver to work on 64 bit machines.

```
Provider=MSDASQL; Driver={MySQL ODBC 5.1  
Driver}; Server=localhost; Database=myDataBase; User=myUsername; Password=myPassword; Opt
```

64 bit version of the driver

Note the "w" in the driver name.

```
Driver={MySQL ODBC 5.2w
Driver}; Server=localhost; Database=myDataBase; User=myUsername; Password=myPassword; Opt
```

Version 5.2.5 new name

MySQL ODBC Connector version 5.2.5 and above register itself as "MySQL ODBC 5.2 Unicode Driver" and "MySQL ODBC 5.2 ANSI Driver".

```
Driver={MySQL ODBC 5.2 ANSI Driver}; Server=localhost; Database=myDataBase;
User=myUsername; Password=myPassword; Option=3;
```

See version 5.2 connection strings [HERE](#)

Remote database

```
Driver={MySQL ODBC 5.1
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo
```

Specifying TCP/IP port

```
Driver={MySQL ODBC 5.1
Driver}; Server=myServerAddress; Port=3306; Database=myDataBase; User=myUsername; Passwor
```

The driver defaults to port value 3306, if not specified in the connection string, as 3306 is the default port for MySQL.

Specifying character set

```
Driver={MySQL ODBC 5.1
Driver}; Server=myServerAddress; charset=UTF8; Database=myDataBase; User=myUsername; Pass
```

Specifying socket

This one specifies the Unix socket file or Windows named pipe to connect to. Used only for local client connections.

```
Driver={MySQL ODBC 5.1
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo
```

On Windows, the socket variable is the name of the named pipe that is used for local client connections. The default value is MySQL.

On Unix platforms, the socket variable is the name of the socket file that is used for local client connections. The default is /tmp/mysql.sock.

Using SSL

```
Driver={MySQL ODBC 5.1
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo
cert.pem; sslkey=c:\client-key.pem; sslverify=1;
Option=3;
```

SSLCA specifies the path to a file with a list of trust SSL CAs

SSLCERT specifies the name of the SSL certificate file to use for establishing a secure connection.

SSLKEY specifies the name of the SSL key file to use for establishing a secure connection.

MySQL Connector/ODBC 3.51

Local database

```
Driver={MySQL ODBC 3.51
Driver}; Server=localhost; Database=myDataBase; User=myUsername; Password=myPassword; Opt
```

Remote database

```
Driver={MySQL ODBC 3.51
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo
```

Specifying TCP/IP port

```
Driver={MySQL ODBC 3.51
Driver}; Server=myServerAddress; Port=3306; Database=myDataBase; User=myUsername; Passwor
```

The driver defaults to port value 3306, if not specified in the connection string, as 3306 is the default port for MySQL.

Specifying character set

```
Driver={MySQL ODBC 3.51
Driver}; Server=myServerAddress; charset=UTF8; Database=myDataBase; User=myUsername; Pass
```

Note that the charset option works from version 3.51.17 of the driver.

Specifying socket

This one specifies the Unix socket file or Windows named pipe to connect to. Used only for local client connections.

```
Driver={MySQL ODBC 3.51
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo
```

On Windows, the socket variable is the name of the named pipe that is used for local client connections. The default value is MySQL.

On Unix platforms, the socket variable is the name of the socket file that is used for local client connections. The default is /tmp/mysql.sock.

Using SSL

```
Driver={MySQL ODBC 3.51
Driver}; Server=myServerAddress; Database=myDataBase; User=myUsername; Password=myPasswo
cert.pem; sslkey=c:\client-key.pem; sslverify=1;
Option=3;
```

SSLCA specifies the path to a file with a list of trust SSL CAs

SSLCERT specifies the name of the SSL certificate file to use for establishing a secure connection.

SSLKEY specifies the name of the SSL key file to use for establishing a secure connection.

MyODBC 2.50

Local database

```
Driver={mySQL}; Server=localhost; Option=16834; Database=myDataBase;
```

Remote database

```
Driver=
{mySQL}; Server=myServerAddress; Option=131072; Stmt=; Database=myDataBase; User=myUsern
```

Specifying TCP/IP port


```
Driver=
{mySQL}; Server=myServerAddress; Port=3306; Option=131072; Stmt=; Database=myDataBase; Us
```

The driver defaults to port value 3306, if not specified in the connection string, as 3306 is the default port for MySQL.

.NET Framework Data Provider for ODBC

Use an ODBC driver from .NET

```
Driver={any odbc driver's name}; OdbcKey1=someValue; OdbcKey2=someValue;
```

See the respective ODBC driver's connection strings options. The .net OdbcConnection will just pass on the connection string to the specified ODBC driver. Read more [here](#) .