SQL Server connection strings

Microsoft SqlClient Data Provider for SQL Server

Standard Security

```
Server=myServerAddress;Database=myDataBase;User Id=myUsername;Password=myPassword;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Trusted Connection

```
Server=myServerAddress;Database=myDataBase;Trusted_Connection=True;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Connection to a SQL Server instance

The ***server/instance*** name syntax used in the ***server*** option is the same for all SQL Server connection strings.

```
Server=myServerName\myInstanceName;Database=myDataBase;User Id=myUsername;Password=myPassword;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Server=myServerName,myPortNumber; Database=myDataBase; User Id=myUsername; Password=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Connect via an IP address

```
Data Source=190.190.200.100,1433; Network Library=DBMSSOCN; Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
```

DBMSSOCN=TCP/IP is how to use TCP/IP instead of Named Pipes. At the end of the Data Source is the port to use. 1433 is the default port for SQL Server. Read more here .

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Enable MARS

```
Server=myServerAddress; Database=myDataBase; Trusted_Connection=True; MultipleActiveResultSets=true;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file on connect to a local SQL Server Express instance

```
Server=.\SQLExpress; AttachDbFilename=C:\MyFolder\MyDataFile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file, located in the data directory, on connect to a local SQL Server Express instance

```
Server=.\SQLExpress; AttachDbFilename=|DataDirectory|mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

User Instance on local SQL Server Express

The ***User Instance*** feature is deprecated with SQL Server 2012, use the ***SQL Server Express LocalDB*** feature instead.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB automatic instance

```
Server=(localdb)\v11.0; Integrated Security=true;
```

The first connection to LocalDB will create and start the instance, this takes some time and might cause a connection timeout failure. If this happens, wait a bit and connect again.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB automatic instance with specific data file

```
Server=(localdb)\v11.0; Integrated Security=true; AttachDbFileName=C:\MyFolder\MyData.mdf;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB named instance

To create a named instance, use the **SqlLocalDB.exe** program. Example `SqlLocalDB.exe create MyInstance` and `SqlLocalDB.exe start MyInstance`

```
Server=(localdb)\MyInstance; Integrated Security=true;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB named instance via the named pipes pipe name

The *Server=(localdb)* syntax is not supported by .NET framework versions before 4.0.2. However the named pipes connection will work to connect pre 4.0.2 applications to LocalDB instances.

```
Server=np:\\.\pipe\LOCALDB#F365A78E\tsql\query;
```

Executing `SqlLocalDB.exe info MyInstance` will get you (along with other info) the instance pipe name such as "np:\\.\pipe\LOCALDB#F365A78E\tsql\query".

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB shared instance

Both automatic and named instances of LocalDB can be shared.

```
Server=(localdb)\.\MyInstanceShare; Integrated Security=true;
```

Use **SqlLocalDB.exe** to share or unshare an instance. For example execute `SqlLocalDB.exe share "MyInstance" "MyInstanceShare"` to share an instance.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Data Source=myServerAddress; Failover Partner=myMirrorServerAddress; Initial Catalog=myDataBase; Integrated Security=True;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Asynchronous processing

A connection to SQL Server that allows for the issuing of async requests through ADO.NET objects.

```
Server=myServerAddress; Database=myDataBase; Integrated Security=True; Asynchronous Processing=True;
```

See also the List of all SqlConnection connection string properties

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Using an User Instance on a local SQL Server Express instance

The User Instance functionality creates a new SQL Server instance on the fly during connect. This works only on a local SQL Server instance and only when connecting using windows authentication over local named pipes. The purpose is to be able to create a full rights SQL Server instance to a user with limited administrative rights on the computer.

```
Data Source=.\SQLExpress; Integrated Security=true; AttachDbFilename=C:\MyFolder\MyDataFile.mdf; User Instance=true;
```

To use the User Instance functionality you need to enable it on the SQL Server. This is done by executing the following command: sp_configure 'user instances enabled', '1'. To disable the functionality execute sp_configure 'user instances enabled', '0'.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2008 SQL Server 2005

Specifying packet size

```
Server=myServerAddress; Database=myDataBase; User ID=myUsername; Password=myPassword; Trusted_Connection=False; Packet Size=4096;
```

By default, the Microsoft .NET Framework Data Provider for SQL Server sets the network packet size to 8192 bytes. This might however not be optimal, try to set this value to 4096 instead. The default value of 8192 might cause ***Failed to reserve contiguous memory*** errors as well, read more here .

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Always Encrypted

```
Data Source=myServer; Initial Catalog=myDB; Integrated Security=true; Column Encryption Setting=enabled;
```

This one is available in .NET Core (as opposed to System.Data.SqlClient).

SQL Server 2019 SQL Server 2017 SQL Server 2016 Azure SQL Database

Always Encrypted with secure enclaves

```
Data Source=myServer; Initial Catalog=myDB; Integrated Security=true; Column Encryption Setting=enabled; Enclave Attestation Url=http://hgs.bastion.loca
```

This one is available in .NET Core (as opposed to System.Data.SqlClient).

SQL Server 2019

Problems connecting?

Get answer in the SQL Server Q & A forum

.NET Framework Data Provider for SQL Server

Standard Security

```
Server=myServerAddress; Database=myDataBase; User Id=myUsername; Password=myPassword;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Trusted Connection

```
Server=myServerAddress; Database=myDataBase; Trusted_Connection=True;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Connection to a SQL Server instance

The ***server/instance*** name syntax used in the ***server*** option is the same for all SQL Server connection strings.

```
Server=myServerName\myInstanceName; Database=myDataBase; User Id=myUsername; Password=myPassword;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Server=myServerName,myPortNumber; Database=myDataBase; User Id=myUsername; Password=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Trusted Connection from a CE device

A Windows CE device is most often not authenticated and logged in to a domain but it is possible to use SSPI or trusted connection and authentication from a CE device using this connection string.

```
Data Source=myServerAddress; Initial Catalog=myDataBase; Integrated Security=SSPI; User ID=myDomain\myUsername; Password=myPassword;
```

Note that this will ***only*** work on a CE device.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Connect via an IP address

```
Data Source=190.190.200.100,1433; Network Library=DBMSSOCN; Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
```

DBMSSOCN=TCP/IP is how to use TCP/IP instead of Named Pipes. At the end of the Data Source is the port to use. 1433 is the default port for SQL Server. Read more here .

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Enable MARS

```
Server=myServerAddress; Database=myDataBase; Trusted_Connection=True; MultipleActiveResultSets=true;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file on connect to a local SQL Server Express instance

```
Server=.\SQLExpress; AttachDbFilename=C:\MyFolder\MyDataFile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file, located in the data directory, on connect to a local SQL Server Express instance

```
Server=.\SQLExpress; AttachDbFilename=|DataDirectory|mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

User Instance on local SQL Server Express

The ***User Instance*** feature is deprecated with SQL Server 2012, use the ***SQL Server Express LocalDB*** feature instead.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB automatic instance

```
Server=(localdb)\v11.0; Integrated Security=true;
```

The first connection to LocalDB will create and start the instance, this takes some time and might cause a connection timeout failure. If this happens, wait a bit and connect again.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB automatic instance with specific data file

```
Server=(localdb)\v11.0; Integrated Security=true; AttachDbFileName=C:\MyFolder\MyData.mdf;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB named instance

To create a named instance, use the **SqlLocalDB.exe** program. Example `SqlLocalDB.exe create MyInstance` and `SqlLocalDB.exe start MyInstance`

```
Server=(localdb)\MyInstance; Integrated Security=true;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB named instance via the named pipes pipe name

The *Server=(localdb)* syntax is not supported by .NET framework versions before 4.0.2. However the named pipes connection will work to connect pre 4.0.2 applications to LocalDB instances.

```
Server=np:\\.\pipe\LOCALDB#F365A78E\tsql\query;
```

Executing `SqlLocalDB.exe info MyInstance` will get you (along with other info) the instance pipe name such as "np:\\.\pipe\LOCALDB#F365A78E\tsql\query".

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

LocalDB shared instance

Both automatic and named instances of LocalDB can be shared.

```
Server=(localdb)\.\MyInstanceShare; Integrated Security=true;
```

Use **SqlLocalDB.exe** to share or unshare an instance. For example execute `SqlLocalDB.exe share "MyInstance" "MyInstanceShare"` to share an instance.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Data Source=myServerAddress; Failover Partner=myMirrorServerAddress; Initial Catalog=myDataBase; Integrated Security=True;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Asynchronous processing

A connection to SQL Server that allows for the issuing of async requests through ADO.NET objects.

```
Server=myServerAddress; Database=myDataBase; Integrated Security=True; Asynchronous Processing=True;
```

See also the List of all SqlConnection connection string properties

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Using an User Instance on a local SQL Server Express instance

The User Instance functionality creates a new SQL Server instance on the fly during connect. This works only on a local SQL Server instance and only when connecting using windows authentication over local named pipes. The purpose is to be able to create a full rights SQL Server instance to a user with limited administrative rights on the computer.

```
Data Source=.\SQLExpress; Integrated Security=true; AttachDbFilename=C:\MyFolder\MyDataFile.mdf; User Instance=true;
```

To use the User Instance functionality you need to enable it on the SQL Server. This is done by executing the following command: sp_configure 'user instances enabled', '1'. To disable the functionality execute sp_configure 'user instances enabled', '0'.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2008 SQL Server 2005

Specifying packet size

```
Server=myServerAddress; Database=myDataBase; User ID=myUsername; Password=myPassword; Trusted_Connection=False; Packet Size=4096;
```

By default, the Microsoft .NET Framework Data Provider for SQL Server sets the network packet size to 8192 bytes. This might however not be optimal, try to set this value to 4096 instead. The default value of 8192 might cause ***Failed to reserve contiguous memory*** errors as well, read more here .

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Always Encrypted

```
Data Source=myServer; Initial Catalog=myDB; Integrated Security=true; Column Encryption Setting=enabled;
```

Always Encrypted in System.Data.SqlClient is available only for .NET Framework, not .NET Core. To use Always Encrypted in .NET Core switch to Microsoft.Data.SqlClient (NuGet-package).

SQL Server 2019 SQL Server 2017 SQL Server 2016 Azure SQL Database

Always Encrypted with secure enclaves

```
Data Source=myServer; Initial Catalog=myDB; Integrated Security=true; Column Encryption Setting=enabled; Enclave Attestation Url=http://hgs.bastion.loca
```

Always Encrypted in System.Data.SqlClient is available only for .NET Framework, not .NET Core. To use Always Encrypted in .NET Core switch to Microsoft.Data.SqlClient (NuGet-package).

SQL Server 2019

Microsoft OLE DB Driver for SQL Server

Standard security

```
Provider=MSOLEDBSQL; Server=myServerAddress; Database=myDataBase; UID=myUsername; PWD=myPassword;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

ADO to map new data types

For ADO to correctly map SQL Server new datatypes, i.e. XML, UDT, varchar(max), nvarchar(max), and varbinary(max), include DataTypeCompatibility=80; in the connection string. If you are not using ADO this is not necessary.

```
Provider=MSOLEDBSQL; DataTypeCompatibility=80; Server=myServerAddress; Database=myDataBase; UID=myUsername; PWD=myPassword;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Trusted connection

```
Provider=MSOLEDBSQL; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Provider=MSOLEDBSQL; Server=myServerName\theInstanceName; Database=myDataBase; Trusted_Connection=yes;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Provider=MSOLEDBSQL; Server=myServerName,myPortNumber; Database=myDataBase; UID=myUsername; PWD=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Enable MARS

```
Provider=MSOLEDBSQL; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; MARS Connection=true;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Encrypt data sent over network

```
Provider=MSOLEDBSQL; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; Encrypt=yes;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Attach a database file on connect to a local SQL Server Express instance

```
Provider=MSOLEDBSQL; Server=.\SQLExpress; AttachDBFilename=c:\asd\qwe\mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Provider=MSOLEDBSQL; Data Source=myServerAddress; Failover Partner=myMirrorServerAddress; Initial Catalog=myDataBase; Integrated Security=True;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Availability group and failover cluster

Enable fast failover for Always On Availability Groups and Failover Cluster Instances. TCP is the only supported protocol. Also set an explicit timeout as these scenarios might require more time.

```
Provider=MSOLEDBSQL; Server=tcp:AvailabilityGroupListenerDnsName,1433; MultiSubnetFailover=Yes; Database=MyDB; Integrated Security=SSPI; Connect Timeo
```

MultiSubnetFailover will perform retries in parallell and do it faster than default TCP retransmit intervals. This can **not** be combined with mirroring, e.g. Failover_Partner=mirrorServer.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Read-Only application intent

Use a read workload when connecting. Enforces read only at connection time, and also for USE database statements.

```
Provider=MSOLEDBSQL; Server=tcp:AvailabilityGroupListenerDnsName,1433; MultiSubnetFailover=Yes; ApplicationIntent=ReadOnly; Database=MyDB; Integrated
```

The result of using ApplicationIntent depends on database configuration. See read-only routing. The default for ApplicationIntent is **ReadWrite**.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Read-Only routing

You can either use an availability group listener for Server **OR** the read-only instance name to enforce a specific read-only instance.

```
Provider=MSOLEDBSQL; Server=aKnownReadOnlyInstance; MultiSubnetFailover=Yes; ApplicationIntent=ReadOnly; Database=MyDB; Integrated Security=SSPI; Conne
```

An availability group must enable read-only routing for this to work.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

SQL Server Native Client 11.0 OLE DB Provider

Standard security

```
Provider=SQLNCLI11; Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

Are you using SQL Server 2012 Express? Don't miss the server name syntax Servername\SQLEXPRESS where you substitute Servername with the name of the computer where the SQL Server 2012 Express installation resides.

When to use SQL Native Client?

SQL Server 2012 SQL Server 2008 SQL Server 2005

Trusted connection

```
Provider=SQLNCLI11; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2012 SQL Server 2008 SQL Server 2005

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Provider=SQLNCLI11; Server=myServerName\theInstanceName; Database=myDataBase; Trusted_Connection=yes;
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Provider=SQLNCLI11; Server=myServerName,myPortNumber; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2012 SQL Server 2008 SQL Server 2005

Prompt for username and password

This one is a bit tricky. First you need to set the connection object's Prompt property to adPromptAlways. Then use the connection string to connect to the database.

```
oConn.Properties("Prompt") = adPromptAlways

oConn.Open "Provider=SQLNCLI11;Server=myServerAddress;DataBase=myDataBase;"
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

Enable MARS

```
Provider=SQLNCLI11; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; MARS Connection=True;
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

Encrypt data sent over network

```
Provider=SQLNCLI11; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; Encrypt=yes;
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file on connect to a local SQL Server Express instance

```
Provider=SQLNCLI11; Server=.\SQLExpress; AttachDbFilename=c:\asd\qwe\mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file, located in the data directory, on connect to a local SQL Server Express instance

```
Provider=SQLNCLI11; Server=.\SQLExpress; AttachDbFilename=|DataDirectory|mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2012 SQL Server 2008 SQL Server 2005

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Provider=SQLNCLI11; Data Source=myServerAddress; Failover Partner=myMirrorServerAddress; Initial Catalog=myDataBase; Integrated Security=True;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

SQL Server 2012 SQL Server 2008 SQL Server 2005

SQL Server Native Client 10.0 OLE DB Provider

Standard security

```
Provider=SQLNCLI10; Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

Are you using SQL Server 2008 Express? Don't miss the server name syntax Servername\SQLEXPRESS where you substitute Servername with the name of the computer where the SQL Server 2008 Express installation resides.

When to use SQL Native Client?

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Trusted connection

```
Provider=SQLNCLI10; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Provider=SQLNCLI10; Server=myServerName\theInstanceName; Database=myDataBase; Trusted_Connection=yes;
```

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Provider=SQLNCLI10; Server=myServerName,myPortNumber; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Prompt for username and password

This one is a bit tricky. First you need to set the connection object's Prompt property to adPromptAlways. Then use the connection string to connect to the database.

```
oConn.Properties("Prompt") = adPromptAlways

oConn.Open "Provider=SQLNCLI10;Server=myServerAddress;DataBase=myDataBase;"
```

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Enable MARS

```
Provider=SQLNCLI10; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; MARS Connection=True;
```

SQL Server 2008 SQL Server 2005

Encrypt data sent over network

```
Provider=SQLNCLI10; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; Encrypt=yes;
```

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Attach a database file on connect to a local SQL Server Express instance

```
Provider=SQLNCLI10; Server=.\SQLExpress; AttachDbFilename=c:\asd\qwe\mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2008 SQL Server 2005

Attach a database file, located in the data directory, on connect to a local SQL Server Express instance

```
Provider=SQLNCLI10; Server=.\SQLExpress; AttachDbFilename=|DataDirectory|mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2008 SQL Server 2005

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Provider=SQLNCLI10; Data Source=myServerAddress; Failover Partner=myMirrorServerAddress; Initial Catalog=myDataBase; Integrated Security=True;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

SQL Server 2008 SQL Server 2005

SQL Native Client 9.0 OLE DB Provider

Standard security

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

Are you using SQL Server 2005 Express? Don't miss the server name syntax Servername\SQLEXPRESS where you substitute Servername with the name of the computer where the SQL Server 2005 Express installation resides.

When to use SQL Native Client?

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Trusted connection

```
Provider=SQLNCLI; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Provider=SQLNCLI; Server=myServerName\theInstanceName; Database=myDataBase; Trusted_Connection=yes;
```

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Provider=SQLNCLI; Server=myServerName,myPortNumber; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Prompt for username and password

This one is a bit tricky. First you need to set the connection object's Prompt property to adPromptAlways. Then use the connection string to connect to the database.

```
oConn.Properties("Prompt") = adPromptAlways

oConn.Open "Provider=SQLNCLI;Server=myServerAddress;DataBase=myDataBase;"
```

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Enable MARS

```
Provider=SQLNCLI; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; MARS Connection=True;
```

SQL Server 2005

Encrypt data sent over network

```
Provider=SQLNCLI; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; Encrypt=yes;
```

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Attach a database file on connect to a local SQL Server Express instance

```
Provider=SQLNCLI; Server=.\SQLExpress; AttachDbFilename=c:\mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2005

Attach a database file, located in the data directory, on connect to a local SQL Server Express instance

```
Provider=SQLNCLI; Server=.\SQLExpress; AttachDbFilename=|DataDirectory|mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2005

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Provider=SQLNCLI; Data Source=myServerAddress; Failover Partner=myMirrorServerAddress; Initial Catalog=myDataBase; Integrated Security=True;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

SQL Server 2005

Microsoft OLE DB Provider for SQL Server

Standard Security

```
Provider=sqloledb; Data Source=myServerAddress; Initial Catalog=myDataBase; User Id=myUsername; Password=myPassword;
```

SQL Server 2000 SQL Server 7.0

Trusted connection

```
Provider=sqloledb; Data Source=myServerAddress; Initial Catalog=myDataBase; Integrated Security=SSPI;
```

Use serverName\instanceName as Data Source to use a specific SQL Server instance. Please note that the multiple SQL Server instances feature is available only from SQL Server version 2000 and not in any previous versions.

SQL Server 2000 SQL Server 7.0

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Provider=sqloledb; Data Source=myServerName\theInstanceName; Initial Catalog=myDataBase; Integrated Security=SSPI;
```

SQL Server 2000 SQL Server 7.0

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Provider=sqloledb; Server=myServerName,myPortNumber; Database=myDataBase; User Id=myUsername; Password=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2000 SQL Server 7.0

Prompt for username and password

This one is a bit tricky. First set the connection object's Provider property to "sqloledb". Thereafter set the connection object's Prompt property to adPromptAlways. Then use the connection string to connect to the database.

```
oConn.Provider = "sqloledb"
oConn.Properties("Prompt") = adPromptAlways

oConn.Open "Data Source=myServerAddress;Initial Catalog=myDataBase;"
```

SQL Server 2000 SQL Server 7.0

Connect via an IP address

```
Provider=sqloledb; Data Source=190.190.200.100,1433; Network Library=DBMSSOCN; Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
```

DBMSSOCN=TCP/IP. This is how to use TCP/IP instead of Named Pipes. At the end of the Data Source is the port to use. 1433 is the default port for SQL Server. Read more in the article How to define which network protocol to use .

SQL Server 2000 SQL Server 7.0

Disable connection pooling

This one is usefull when receving errors "sp_setapprole was not invoked correctly." (7.0) or "General network error. Check your network documentation" (2000) when connecting using an application role enabled connection. Application pooling (or OLE DB resource pooling) is on by default. Disabling it can help on this error.

```
Provider=sqloledb; Data Source=myServerAddress; Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword; OLE DB Services=-2;
```

SQL Server 2000 SQL Server 7.0

.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 .

Microsoft ODBC Driver 17 for SQL Server

Standard security

```
Driver={ODBC Driver 17 for SQL Server}; Server=myServerAddress; Database=myDataBase; UID=myUsername; PWD=myPassword;
```

Using SQL Server Express? The server name syntax is ServerName\SQLEXPRESS where you substitute ServerName with the name of the server where SQL Server Express is running.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Trusted Connection

```
Driver={ODBC Driver 17 for SQL Server}; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Driver={ODBC Driver 17 for SQL Server}; Server=serverName\instanceName; Database=myDataBase; Trusted_Connection=yes;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Driver={ODBC Driver 17 for SQL Server}; Server=myServerName,myPortNumber; Database=myDataBase; UID=myUsername; PWD=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Enable MARS

```
Driver={ODBC Driver 17 for SQL Server}; Server=serverAddress; Database=databaseName; Trusted_Connection=yes; MARS_Connection=yes;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Encrypt data sent over network

```
Driver={ODBC Driver 17 for SQL Server}; Server=serverAddress; Database=databaseName; Trusted_Connection=yes; Encrypt=yes;
```

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Attach a database file on connect to a local SQL Server Express instance

```
Driver={ODBC Driver 17 for SQL Server}; Server=.\SQLExpress; AttachDBFileName=c:\dir\\mydb.mdf; Database=dbName; Trusted_Connection=yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Database mirroring

If you connect to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Driver={ODBC Driver 17 for SQL
Server}; Server=myServerAddress; Failover_Partner=myMirrorServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

This one is working only on Windows, not on macOS or Linux. There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

Please note if you are using TCP/IP (using the network library parameter) and database mirroring, including port number in the address (formed as servername,portnumber) for both the main server and the failover partner can solve some reported issues.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Microsoft ODBC Driver 13 for SQL Server

Standard security

```
Driver={ODBC Driver 13 for SQL Server}; Server=myServerAddress; Database=myDataBase; UID=myUsername; PWD=myPassword;
```

Using SQL Server Express? The server name syntax is ServerName\SQLEXPRESS where you substitute ServerName with the name of the server where SQL Server Express is running.

SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Trusted Connection

```
Driver={ODBC Driver 13 for SQL Server}; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Driver={ODBC Driver 13 for SQL Server};Server=serverName\instanceName;Database=myDataBase;Trusted_Connection=yes;
```

SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Driver={ODBC Driver 13 for SQL Server};Server=myServerName,myPortNumber;Database=myDataBase;UID=myUsername;PWD=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Enable MARS

```
Driver={ODBC Driver 13 for SQL Server};Server=serverAddress;Database=databaseName;Trusted_Connection=yes;MARS_Connection=yes;
```

SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Encrypt data sent over network

```
Driver={ODBC Driver 13 for SQL Server};Server=serverAddress;Database=databaseName;Trusted_Connection=yes;Encrypt=yes;
```

SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Attach a database file on connect to a local SQL Server Express instance

```
Driver={ODBC Driver 13 for SQL Server};Server=.\SQLExpress;AttachDBFileName=c:\dir\\mydb.mdf;Database=dbName;Trusted_Connection=yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Database mirroring

If you connect to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Driver={ODBC Driver 13 for SQL
Server};Server=myServerAddress;Failover_Partner=myMirrorServerAddress;Database=myDataBase;Trusted_Connection=yes;
```

This one is working only on Windows, not on macOS or Linux. There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

Please note if you are using TCP/IP (using the network library parameter) and database mirroring, including port number in the address (formed as servername,portnumber) for both the main server and the failover partner can solve some reported issues.

SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012 SQL Server 2008

Microsoft ODBC Driver 11 for SQL Server

Standard security

```
Driver={ODBC Driver 11 for SQL Server};Server=myServerAddress;Database=myDataBase;UID=myUsername;PWD=myPassword;
```

Using SQL Server Express? The server name syntax is ServerName\SQLEXPRESS where you substitute ServerName with the name of the server where SQL Server Express is running.

SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Trusted Connection

```
Driver={ODBC Driver 11 for SQL Server};Server=myServerAddress;Database=myDataBase;Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Driver={ODBC Driver 11 for SQL Server};Server=serverName\instanceName;Database=myDataBase;Trusted_Connection=yes;
```

SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Driver={ODBC Driver 11 for SQL Server}; Server=myServerName,myPortNumber; Database=myDataBase; UID=myUsername; PWD=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Enable MARS

```
Driver={ODBC Driver 11 for SQL Server}; Server=serverAddress; Database=databaseName; Trusted_Connection=yes; MARS_Connection=yes;
```

SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Encrypt data sent over network

```
Driver={ODBC Driver 11 for SQL Server}; Server=serverAddress; Database=databaseName; Trusted_Connection=yes; Encrypt=yes;
```

SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file on connect to a local SQL Server Express instance

```
Driver={ODBC Driver 11 for SQL Server}; Server=.\SQLExpress; AttachDBFileName=c:\dir\\mydb.mdf; Database=dbName; Trusted_Connection=yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

Database mirroring

If you connect to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Driver={ODBC Driver 11 for SQL
Server}; Server=myServerAddress; Failover_Partner=myMirrorServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

This one is working only on Windows, not on macOS or Linux. There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

Please note if you are using TCP/IP (using the network library parameter) and database mirroring, including port number in the address (formed as servername,portnumber) for both the main server and the failover partner can solve some reported issues.

SQL Server 2014 SQL Server 2012 SQL Server 2008 SQL Server 2005

SQL Server Native Client 11.0 ODBC Driver

Standard security

```
Driver={SQL Server Native Client 11.0}; Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

Are you using SQL Server 2012 Express? Don't miss the server name syntax Servername\SQLEXPRESS where you substitute Servername with the name of the computer where the SQL Server 2012 Express installation resides.

When to use SQL Native Client?

SQL Server 2012 SQL Server 2008 SQL Server 2005

Trusted Connection

```
Driver={SQL Server Native Client 11.0}; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2012 SQL Server 2008 SQL Server 2005

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Driver={SQL Server Native Client 11.0}; Server=myServerName\theInstanceName; Database=myDataBase; Trusted_Connection=yes;
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Driver={SQL Server Native Client 11.0};Server=myServerName,myPortNumber;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2012 SQL Server 2008 SQL Server 2005

Prompt for username and password

This one is a bit tricky. First you need to set the connection object's Prompt property to adPromptAlways. Then use the connection string to connect to the database.

```
oConn.Properties("Prompt") = adPromptAlways

oConn.Open "Driver={SQL Server Native Client 11.0};Server=myServerAddress;Database=myDataBase;"
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

Enable MARS

```
Driver={SQL Server Native Client 11.0};Server=myServerAddress;Database=myDataBase;Trusted_Connection=yes;MARS_Connection=yes;
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

Encrypt data sent over network

```
Driver={SQL Server Native Client 11.0};Server=myServerAddress;Database=myDataBase;Trusted_Connection=yes;Encrypt=yes;
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file on connect to a local SQL Server Express instance

```
Driver={SQL Server Native Client 11.0};Server=.\SQLExpress;AttachDbFilename=c:\asd\qwe\mydbfile.mdf;Database=dbname;Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2012 SQL Server 2008 SQL Server 2005

Attach a database file, located in the data directory, on connect to a local SQL Server Express instance

```
Driver={SQL Server Native Client
11.0};Server=.\SQLExpress;AttachDbFilename=|DataDirectory|mydbfile.mdf;Database=dbname;Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2012 SQL Server 2008 SQL Server 2005

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Driver={SQL Server Native Client
11.0};Server=myServerAddress;Failover_Partner=myMirrorServerAddress;Database=myDataBase;Trusted_Connection=yes;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

Please note if you are using TCP/IP (using the network library parameter) and database mirroring, including port number in the address (formed as servername,portnumber) for both the main server and the failover partner can solve some reported issues.

SQL Server 2012 SQL Server 2008 SQL Server 2005

SQL Server Native Client 10.0 ODBC Driver

Standard security

```
Driver={SQL Server Native Client 10.0};Server=myServerAddress;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
```

Are you using SQL Server 2008 Express? Don't miss the server name syntax Servername\SQLEXPRESS where you substitute Servername with the name of the computer where the SQL Server 2008 Express installation resides.

When to use SQL Native Client?

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Trusted Connection

```
Driver={SQL Server Native Client 10.0};Server=myServerAddress;Database=myDataBase;Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Driver={SQL Server Native Client 10.0};Server=myServerName\theInstanceName;Database=myDataBase;Trusted_Connection=yes;
```

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Driver={SQL Server Native Client 10.0};Server=myServerName,myPortNumber;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Prompt for username and password

This one is a bit tricky. First you need to set the connection object's Prompt property to adPromptAlways. Then use the connection string to connect to the database.

```
oConn.Properties("Prompt") = adPromptAlways

oConn.Open "Driver={SQL Server Native Client 10.0};Server=myServerAddress;Database=myDataBase;"
```

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Enable MARS

```
Driver={SQL Server Native Client 10.0};Server=myServerAddress;Database=myDataBase;Trusted_Connection=yes;MARS_Connection=yes;
```

SQL Server 2008 SQL Server 2005

Encrypt data sent over network

```
Driver={SQL Server Native Client 10.0};Server=myServerAddress;Database=myDataBase;Trusted_Connection=yes;Encrypt=yes;
```

SQL Server 2008 SQL Server 2005 SQL Server 2000 SQL Server 7.0

Attach a database file on connect to a local SQL Server Express instance

```
Driver={SQL Server Native Client 10.0};Server=.\SQLExpress;AttachDbFilename=c:\asd\qwe\mydbfile.mdf;Database=dbname;Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2008 SQL Server 2005

Attach a database file, located in the data directory, on connect to a local SQL Server Express instance

```
Driver={SQL Server Native Client
10.0};Server=.\SQLExpress;AttachDbFilename=|DataDirectory|mydbfile.mdf;Database=dbname;Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2008 SQL Server 2005

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Driver={SQL Server Native Client
10.0};Server=myServerAddress;Failover_Partner=myMirrorServerAddress;Database=myDataBase;Trusted_Connection=yes;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

Please note if you are using TCP/IP (using the network library parameter) and database mirroring, including port number in the address (formed as servername,portnumber) for both the main server and the failover partner can solve some reported issues.

SQL Server 2008 SQL Server 2005

SQL Native Client 9.0 ODBC Driver

Standard security

```
Driver={SQL Native Client}; Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

Are you using SQL Server 2005 Express? Don't miss the server name syntax Servername\SQLEXPRESS where you substitute Servername with the name of the computer where the SQL Server 2005 Express installation resides.

When to use SQL Native Client?

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Trusted Connection

```
Driver={SQL Native Client}; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

Equivalent key-value pair: "Integrated Security=SSPI" equals "Trusted_Connection=yes"

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Connecting to an SQL Server instance

The syntax of specifying the server instance in the value of the server key is the same for all connection strings for SQL Server.

```
Driver={SQL Native Client}; Server=myServerName\theInstanceName; Database=myDataBase; Trusted_Connection=yes;
```

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Driver={SQL Native Client}; Server=myServerName,myPortNumber; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Prompt for username and password

This one is a bit tricky. First you need to set the connection object's Prompt property to adPromptAlways. Then use the connection string to connect to the database.

```
oConn.Properties("Prompt") = adPromptAlways

oConn.Open "Driver={SQL Native Client};Server=myServerAddress;Database=myDataBase;"
```

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Enable MARS

```
Driver={SQL Native Client}; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; MARS_Connection=yes;
```

SQL Server 2005

Encrypt data sent over network

```
Driver={SQL Native Client}; Server=myServerAddress; Database=myDataBase; Trusted_Connection=yes; Encrypt=yes;
```

SQL Server 2005 SQL Server 2000 SQL Server 7.0

Attach a database file on connect to a local SQL Server Express instance

```
Driver={SQL Native Client}; Server=.\SQLExpress; AttachDbFilename=c:\mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2005

Attach a database file, located in the data directory, on connect to a local SQL Server Express instance

```
Driver={SQL Native Client}; Server=.\SQLExpress; AttachDbFilename=|DataDirectory|mydbfile.mdf; Database=dbname; Trusted_Connection=Yes;
```

Why is the Database parameter needed? If the named database have already been attached, SQL Server does not reattach it. It uses the attached database as the default for the connection.

SQL Server 2005

Database mirroring

If you connect with ADO.NET or the SQL Native Client to a database that is being mirrored, your application can take advantage of the drivers ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string and the failover partner server.

```
Driver={SQL Server Native Client
10.0}; Server=myServerAddress; Failover_Partner=myMirrorServerAddress; Database=myDataBase; Trusted_Connection=yes;
```

There is ofcourse many other ways to write the connection string using database mirroring, this is just one example pointing out the failover functionality. You can combine this with the other connection strings options available.

Please note if you are using TCP/IP (using the network library parameter) and database mirroring, including port number in the address (formed as servername,portnumber) for both the main server and the failover partner can solve some reported issues.

SQL Server 2005

Microsoft SQL Server ODBC Driver

Standard Security

```
Driver={SQL Server}; Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

SQL Server 2000 SQL Server 7.0

Trusted connection

```
Driver={SQL Server}; Server=myServerAddress; Database=myDataBase; Trusted_Connection=Yes;
```

SQL Server 2000 SQL Server 7.0

Using a non-standard port

If your SQL Server listens on a non-default port you can specify that using the servername,xxxx syntax (note the comma, it's not a colon).

```
Driver={SQL Server}; Server=myServerName,myPortNumber; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

The default SQL Server port is 1433 and there is no need to specify that in the connection string.

SQL Server 2000 SQL Server 7.0

Prompt for username and password

This one is a bit tricky. First you need to set the connection object's Prompt property to adPromptAlways. Then use the connection string to connect to the database.

```
oConn.Properties("Prompt") = adPromptAlways

oConn.Open "Driver={SQL Server};Server=myServerAddress;Database=myDataBase;"
```

SQL Server 2000 SQL Server 7.0

.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 .

SQLXML 4.0 OLEDB Provider

With Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)

The DataTypeCompatibility=80 is important for the XML types to be recognised by ADO.

```
Provider=SQLXMLOLEDB.4.0; Data Provider=MSOLEDBSQL; DataTypeCompatibility=80; Data Source=myServerAddress; Initial Catalog=myDataBase; User Id=myUserr
```

See also the other options available for MSOLEDBSQL connection strings.

SQL Server 2019 SQL Server 2017 SQL Server 2016 SQL Server 2014 SQL Server 2012

Using SQL Server Native Client provider 11 (SQLNCLI11)

```
Provider=SQLXMLOLEDB.4.0; Data Provider=SQLNCLI11; Data Source=myServerAddress; Initial Catalog=myDataBase; User Id=myUsername; Password=myPassword;
```

SQL Server 2012

Using SQL Server Native Client provider 10 (SQLNCLI10)

```
Provider=SQLXMLOLEDB.4.0;Data Provider=SQLNCLI10;Data Source=myServerAddress;Initial Catalog=myDataBase;User Id=myUsername;Password=myPassword;
```

SQL Server 2008

Using SQL Server Native Client provider (SQLNCLI)

```
Provider=SQLXMLOLEDB.4.0;Data Provider=SQLNCLI;Data Source=myServerAddress;Initial Catalog=myDataBase;User Id=myUsername;Password=myPassword;
```

SQL Server 2005

SQLXML 3.0 OLEDB Provider

Using SQL Server Ole Db

The SQLXML version 3.0 restricts the data provider to SQLOLEDB only.

```
Provider=SQLXMLOLEDB.3.0;Data Provider=SQLOLEDB;Data Source=myServerAddress;Initial Catalog=myDataBase;User Id=myUsername;Password=myPassword;
```

SQL Server 2000 SQL Server 7.0

Context Connection

Context Connection

Connecting to "self" from within your CLR stored prodedure/function. The context connection lets you execute Transact-SQL statements in the same context (connection) that your code was invoked in the first place.

```
C#
 using(SqlConnection connection = new SqlConnection("context connection=true"))
 {
     connection.Open();
     // Use the connection
 }

VB.Net
 Using connection as new SqlConnection("context connection=true")
     connection.Open()
     ' Use the connection
 End Using
```

SQL Server 2012 SQL Server 2008 SQL Server 2005

MSDataShape

MSDataShape

```
Provider=MSDataShape;Data Provider=SQLOLEDB;Data Source=myServerAddress;Initial Catalog=myDataBase;User ID=myUsername;Password=myPassword;
```

See also the List of all SqlConnection connection string properties

SQL Server 2000 SQL Server 7.0

Connect to

SQL Server ×157

SQL Server 2019 ×64

SQL Server 2017 ×70
SQL Server 2016 ×70
SQL Server 2014 ×76
SQL Server 2012 ×96
SQL Server 2008 ×93
SQL Server 2005 ×97
SQL Server 2000 ×52
SQL Server 7.0 ×52

Sponsored by

Questions

ask question »

SQL Server Does not Exist

Not able to connect from Azure function app to SQL DB on Azure VM

Azure SQL DB connection string is not working unless logged in to azure account

Making Constant Data Bases to recall in program.

Connection error IIS10/SQL 2016

Articles

view all »

The new Microsoft.Data.SqlClient explained

Application Name for SQL Server Connections

All SQL Server SqlConnection Properties

SQL Server Data Types Reference

Network Protocol for SQL Server Connection

When to use the SQL Native Client

Download SQL Server Native Client

SQL Server 2000 Data Types Reference

SQL Server 2005 Data Types Reference

SQL Server 2008 Data Types Reference