This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

.NET Framework Class Library

**SqlConnection.ConnectionString Property**

Gets or sets the string used to open a SQL Server database.

**Namespace:** [System.Data.SqlClient](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.aspx)  
**Assembly:** System.Data (in System.Data.dll)

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifSyntax

Visual Basic

<SettingsBindableAttribute(True)> \_

Public Overrides Property ConnectionString As String

C#

[SettingsBindableAttribute(true)]

public override string ConnectionString { get; set; }

Visual C++

[SettingsBindableAttribute(true)]

public:

virtual property String^ ConnectionString {

String^ get () override;

void set (String^ value) override;

}

F#

[<SettingsBindableAttribute(true)>]

abstract ConnectionString : string with get, set

[<SettingsBindableAttribute(true)>]

override ConnectionString : string with get, set

**Property Value**

Type: [System.String](http://msdn.microsoft.com/en-us/library/system.string.aspx)  
The connection string that includes the source database name, and other parameters needed to establish the initial connection. The default value is an empty string.

**Implements**

[IDbConnection.ConnectionString](http://msdn.microsoft.com/en-us/library/system.data.idbconnection.connectionstring.aspx)

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifExceptions

|  |  |
| --- | --- |
| **Exception** | **Condition** |
| [ArgumentException](http://msdn.microsoft.com/en-us/library/system.argumentexception.aspx) | An invalid connection string argument has been supplied, or a required connection string argument has not been supplied. |

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifRemarks

The **ConnectionString** is similar to an OLE DB connection string, but is not identical. Unlike OLE DB or ADO, the connection string that is returned is the same as the user-set **ConnectionString**, minus security information if the Persist Security Info value is set to **false** (default). The .NET Framework Data Provider for SQL Server does not persist or return the password in a connection string unless you set Persist Security Info to **true**.

You can use the **ConnectionString** property to connect to a database. The following example illustrates a typical connection string.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl00_code');" \o "Copy Code)

"Persist Security Info=False;Integrated Security=true;Initial Catalog=Northwind;server=(local)"

Use the new [SqlConnectionStringBuilder](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnectionstringbuilder.aspx) to construct valid connection strings at run time. For more information, see [Connection String Builders (ADO.NET)](http://msdn.microsoft.com/en-us/library/ms254947.aspx).

The **ConnectionString** property can be set only when the connection is closed. Many of the connection string values have corresponding read-only properties. When the connection string is set, these properties are updated, except when an error is detected. In this case, none of the properties are updated. [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) properties return only those settings that are contained in the **ConnectionString**.

To connect to a local computer, specify "(local)" for the server. If a server name is not specified, a connection will be attempted to the default instance on the local computer.

Resetting the **ConnectionString** on a closed connection resets all connection string values (and related properties) including the password. For example, if you set a connection string that includes "Database= AdventureWorks", and then reset the connection string to "Data Source=myserver;Integrated Security=true", the [Database](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.database.aspx) property is no longer set to "AdventureWorks".

The connection string is parsed immediately after being set. If errors in syntax are found when parsing, a runtime exception, such as [ArgumentException](http://msdn.microsoft.com/en-us/library/system.argumentexception.aspx), is generated. Other errors can be found only when an attempt is made to open the connection.

The basic format of a connection string includes a series of keyword/value pairs separated by semicolons. The equal sign (=) connects each keyword and its value. To include values that contain a semicolon, single-quote character, or double-quote character, the value must be enclosed in double quotation marks. If the value contains both a semicolon and a double-quote character, the value can be enclosed in single quotation marks. The single quotation mark is also useful if the value starts with a double-quote character. Conversely, the double quotation mark can be used if the value starts with a single quotation mark. If the value contains both single-quote and double-quote characters, the quotation mark character used to enclose the value must be doubled every time it occurs within the value.

To include preceding or trailing spaces in the string value, the value must be enclosed in either single quotation marks or double quotation marks. Any leading or trailing spaces around integer, Boolean, or enumerated values are ignored, even if enclosed in quotation marks. However, spaces within a string literal keyword or value are preserved. Using .NET Framework version 1.1 or later, single or double quotation marks may be used within a connection string without using delimiters (for example, Data Source= my'Server or Data Source= my"Server), unless a quotation mark character is the first or last character in the value.

Keywords are not case sensitive.

The following table lists the valid names for keyword values within the **ConnectionString**.

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Default** | **Description** |
| Addr | N/A | Synonym of **Data Source**. |
| Address | N/A | Synonym of **Data Source**. |
| App | N/A | Synonym of **Application Name**. |
| Application Name | N/A | The name of the application, or '.NET SQLClient Data Provider' if no application name is provided.  An application name can be 128 characters or less. |
| **ApplicationIntent** | **ReadWrite** | Declares the application workload type when connecting to a server. Possible values are **ReadOnly** and **ReadWrite**. For example:  [Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl06_code');" \o "Copy Code)  ApplicationIntent=ReadOnly  For more information about SqlClient support for Always On Availability Groups, see [SqlClient Support for High Availability, Disaster Recovery](http://msdn.microsoft.com/en-us/library/hh205662.aspx). |
| Asynchronous Processing  -or-  Async | 'false' | When **true**, enables asynchronous operation support. Recognized values are **true**, **false**, **yes**, and **no**. |
| AttachDBFilename  -or-  Extended Properties  -or-  Initial File Name | N/A | The name of the primary database file, including the full path name of an attachable database. AttachDBFilename is only supported for primary data files with an .mdf extension.  If the value of the AttachDBFileName key is specified in the connection string, the database is attached and becomes the default database for the connection.  If this key is not specified and if the database was previously attached, the database will not be reattached. The previously attached database will be used as the default database for the connection.  If this key is specified together with the AttachDBFileName key, the value of this key will be used as the alias. However, if the name is already used in another attached database, the connection will fail.  The path may be absolute or relative by using the DataDirectory substitution string. If DataDirectory is used, the database file must exist within a subdirectory of the directory pointed to by the substitution string.  Description: Note**Note**  Remote server, HTTP, and UNC path names are not supported.  The database name must be specified with the keyword 'database' (or one of its aliases) as in the following:  "AttachDbFileName=|DataDirectory|\data\YourDB.mdf;integrated security=true;database=YourDatabase"  An error will be generated if a log file exists in the same directory as the data file and the 'database' keyword is used when attaching the primary data file. In this case, remove the log file. Once the database is attached, a new log file will be automatically generated based on the physical path. |
| Connection Lifetime  -or-  Load Balance Timeout | 0 | When a connection is returned to the pool, its creation time is compared with the current time, and the connection is destroyed if that time span (in seconds) exceeds the value specified by **Connection Lifetime**. This is useful in clustered configurations to force load balancing between a running server and a server just brought online.  A value of zero (0) causes pooled connections to have the maximum connection timeout. |
| Connect Timeout  -or-  Connection Timeout  -or-  Timeout | 15 | The length of time (in seconds) to wait for a connection to the server before terminating the attempt and generating an error.  Valid values are greater than or equal to 0 and less than or equal to 2147483647. |
| Context Connection | 'false' | **true** if an in-process connection to SQL Server should be made. |
| Current Language  -or-  Language | N/A | Sets the language used for database server warning or error messages.  The language name can be 128 characters or less. |
| Data Source  -or-  Server  -or-  Address  -or-  Addr  -or-  Network Address | N/A | The name or network address of the instance of SQL Server to which to connect. The port number can be specified after the server name:  server=tcp:servername, portnumber  When specifying a local instance, always use (local). To force a protocol, add one of the following prefixes:  np:(local), tcp:(local), lpc:(local)  **Data Source** must use the TCP format or the Named Pipes format.  TCP format is as follows:   * tcp:<host name>\<instance name> * tcp:<host name>,<TCP/IP port number>   The TCP format must start with the prefix "tcp:" and is followed by the database instance, as specified by a host name and an instance name.  The host name MUST be specified in one of the following ways:   * NetBIOSName * IPv4Address * IPv6Address   The instance name is used to resolve to a particular TCP/IP port number on which a database instance is hosted. Alternatively, specifying a TCP/IP port number directly is also allowed. If both instance name and port number are not present, the default database instance is used.  The Named Pipes format is as follows:   * np:\\<host name>\pipe\<pipe name>   The Named Pipes format MUST start with the prefix "np:" and is followed by a named pipe name.  The host name MUST be specified in one of the following ways:   * NetBIOSName * IPv4Address * IPv6Address   The pipe name is used to identify the database instance to which the .NET Framework application will be connected.  If the value of the **Network** key is specified, the prefixes "tcp:" and "np:" should not be specified.  Description: Note**Note**  ADO.NET 2.0 does not support asynchronous commands over shared memory for SQL Server 2000 or earlier. However, you can force the use of TCP instead of shared memory, either by prefixing **tcp:** to the server name in the connection string, or by using **localhost**. |
| Encrypt | 'false' | When **true**, SQL Server uses SSL encryption for all data sent between the client and server if the server has a certificate installed. Recognized values are **true**, **false**, **yes**, and **no**. |
| Enlist | 'true' | **true** indicates that the SQL Server connection pooler automatically enlists the connection in the creation thread's current transaction context. |
| Failover Partner | N/A | The name of the failover partner server where database mirroring is configured.  If the value of this key is "", then **Initial Catalog** must be present, and its value must not be "".  The server name can be 128 characters or less.  If you specify a failover partner but the failover partner server is not configured for database mirroring and the primary server (specified with the Server keyword) is not available, then the connection will fail.  If you specify a failover partner and the primary server is not configured for database mirroring, the connection to the primary server (specified with the Server keyword) will succeed if the primary server is available.  The Failover Partner keyword is not supported by .NET Framework version 1.0 or 1.1. |
| Initial Catalog  -or-  Database | N/A | The name of the database.  The database name can be 128 characters or less. |
| Integrated Security  -or-  Trusted\_Connection | 'false' | When **false**, User ID and Password are specified in the connection. When **true**, the current Windows account credentials are used for authentication.  Recognized values are **true**, **false**, **yes**, **no**, and **sspi** (strongly recommended), which is equivalent to **true**.  If User ID and Password are specified and Integrated Security is set to true, the User ID and Password will be ignored and Integrated Security will be used. |
| Max Pool Size | 100 | The maximum number of connections that are allowed in the pool.  Valid values are greater than or equal to 1. Values that are less than **Min Pool Size** generate an error. |
| Min Pool Size | 0 | The minimum number of connections that are allowed in the pool.  Valid values are greater than or equal to 0. Zero (0) in this field means no minimum connections are initially opened.  Values that are greater than **Max Pool Size** generate an error. |
| MultipleActiveResultSets | 'false' | When **true**, an application can maintain multiple active result sets (MARS). When **false**, an application must process or cancel all result sets from one batch before it can execute any other batch on that connection.  Recognized values are **true** and **false**.  The keyword is not supported by .NET Framework version 1.0 or 1.1. |
| **MultiSubnetFailover** | FALSE | If your application is connecting to a high-availability, disaster recovery (Always On) availability group (AG), setting **MultiSubnetFailover=Yes** configures SqlClient to provide faster detection of and connection to the (currently) active server. Possible values are **Yes** and **No**, **True** and **False** or **1** and **0**. For example:  [Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl10_code');" \o "Copy Code)  MultiSubnetFailover=Yes  The default is **False**. For more information about SqlClient's support for Always On AGs, see [SqlClient Support for High Availability, Disaster Recovery](http://msdn.microsoft.com/en-us/library/hh205662.aspx). |
| Network Library  -or-  Network  -or-  Net | N/A | The network library used to establish a connection to an instance of SQL Server. Supported values include:  dbnmpntw (Named Pipes)  dbmsrpcn (Multiprotocol, Windows RPC)  dbmsadsn (Apple Talk)  dbmsgnet (VIA)  dbmslpcn (Shared Memory)  dbmsspxn (IPX/SPX)  dbmssocn (TCP/IP)  Dbmsvinn (Banyan Vines)  The corresponding network DLL must be installed on the system to which you connect. If you do not specify a network and you use a local server (for example, "." or "(local)"), shared memory is used. In this example, the network library is Win32 Winsock TCP/IP (dbmssocn), and 1433 is the port being used.  [Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl12_code');" \o "Copy Code)  Network Library=dbmssocn;Data Source=000.000.000.000,1433; |
| Packet Size | 8192 | Size in bytes of the network packets used to communicate with an instance of SQL Server.  The packet size can be greater than or equal to 512 and less than or equal to 32768. |
| Password  -or-  PWD | N/A | The password for the SQL Server account logging on. Not recommended. To maintain a high level of security, we strongly recommend that you use the **Integrated Security** or **Trusted\_Connection** keyword instead.  The password must be 128 characters or less. |
| Persist Security Info  -or-  PersistSecurityInfo | 'false' | When set to **false** or **no** (strongly recommended), security-sensitive information, such as the password, is not returned as part of the connection if the connection is open or has ever been in an open state. Resetting the connection string resets all connection string values including the password. Recognized values are **true**, **false**, **yes**, and **no**. |
| Pooling | 'true' | When the value of this key is set to true, any newly created connection will be added to the pool when closed by the application. In a next attempt to open the same connection, that connection will be drawn from the pool.  Connections are considered the same if they have the same connection string. Different connections have different connection strings.  The value of this key can be "true", "false", "yes", or "no". |
| Replication | 'false' | **true** if replication is supported using the connection. |
| Transaction Binding | Implicit Unbind | Controls connection association with an enlisted **System.Transactions** transaction.  Possible values are:  Transaction Binding=Implicit Unbind;  Transaction Binding=Explicit Unbind;  Implicit Unbind causes the connection to detach from the transaction when it ends. After detaching, additional requests on the connection are performed in autocommit mode. The **System.Transactions.Transaction.Current** property is not checked when executing requests while the transaction is active. After the transaction has ended, additional requests are performed in autocommit mode.  If the system ends the transaction (in the scope of a using block) before the last command completes, it will throw [InvalidOperationException](http://msdn.microsoft.com/en-us/library/system.invalidoperationexception.aspx).  Explicit Unbind causes the connection to remain attached to the transaction until the connection is closed or an explicit **SqlConnection.TransactionEnlist(null)** is called. Beginning in .NET Framework version 4, changes to Implicit Unbind make Explicit Unbind obsolete. An **InvalidOperationException** is thrown if **Transaction.Current** is not the enlisted transaction or if the enlisted transaction is not active. |
| TrustServerCertificate | 'false' | When set to **true**, SSL is used to encrypt the channel when bypassing walking the certificate chain to validate trust. If TrustServerCertificate is set to **true** and Encrypt is set to **false**, the channel is not encrypted. Recognized values are **true**, **false**, **yes**, and **no**. For more information, see [Connection String Syntax (ADO.NET)](http://msdn.microsoft.com/en-us/library/ms254500.aspx). |
| Type System Version | N/A | A string value that indicates the type system the application expects. Possible values are:  Type System Version=SQL Server 2000;  Type System Version=SQL Server 2005;  Type System Version=SQL Server 2008;  Type System Version=Latest;  When set to **SQL Server 2000**, the SQL Server 2000 type system is used. The following conversions are performed when connecting to a SQL Server 2005 instance:  XML to NTEXT  UDT to VARBINARY  VARCHAR(MAX), NVARCHAR(MAX) and VARBINARY(MAX) to TEXT, NEXT and IMAGE respectively.  When set to **SQL Server 2005**, the SQL Server 2005 type system is used. No conversions are made for the current version of ADO.NET.  When set to **Latest**, the latest version than this client-server pair can handle is used. This will automatically move forward as the client and server components are upgraded. |
| User ID  -or-  UID  -or- | N/A | The SQL Server login account. Not recommended. To maintain a high level of security, we strongly recommend that you use the **Integrated Security** or **Trusted\_Connection** keywords instead.  The user ID must be 128 characters or less. |
| User Instance | 'false' | A value that indicates whether to redirect the connection from the default SQL Server Express instance to a runtime-initiated instance running under the account of the caller. |
| Workstation ID  -or-  WSID | The local computer name | The name of the workstation connecting to SQL Server.  The ID must be 128 characters or less. |

The following list contains the valid names for connection pooling values within the **ConnectionString**. For more information, see [SQL Server Connection Pooling (ADO.NET)](http://msdn.microsoft.com/en-us/library/8xx3tyca.aspx).

* Connection Lifetime (or Load Balance Timeout)
* Enlist
* Max Pool Size
* Min Pool Size
* Pooling

When you are setting keyword or connection pooling values that require a Boolean value, you can use 'yes' instead of 'true', and 'no' instead of 'false'. Integer values are represented as strings.

|  |
| --- |
| **Description: NoteNote** |
| The .NET Framework Data Provider for SQL Server uses its own protocol to communicate with SQL Server. Therefore, it does not support the use of an ODBC data source name (DSN) when connecting to SQL Server because it does not add an ODBC layer. |
| **Description: NoteNote** |
| Universal data link (UDL) files are not supported for the .NET Framework Data Provider for SQL Server. |

|  |
| --- |
| **Description: Caution noteCaution** |
| In this release, the application should use caution when constructing a connection string based on user input (for example when retrieving user ID and password information from a dialog box, and appending it to the connection string). The application should make sure that a user cannot embed additional connection string parameters in these values (for example, entering a password as "validpassword;database=somedb" in an attempt to attach to a different database). If you need to construct connection strings based on user input, use the new [SqlConnectionStringBuilder](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnectionstringbuilder.aspx), which validates the connection string and helps to eliminate this problem. See [Connection String Builders (ADO.NET)](http://msdn.microsoft.com/en-us/library/ms254947.aspx) for more information. |

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifExamples

The following example creates a [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) and sets the **ConnectionString** property before opening the connection.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl58_ctl00_ctl01_code');" \o "Copy Code)

Private Sub OpenSqlConnection()

Dim connectionString As String = GetConnectionString()

Using connection As New SqlConnection()

connection.ConnectionString = connectionString

connection.Open()

Console.WriteLine("State: {0}", connection.State)

Console.WriteLine("ConnectionString: {0}", \_

connection.ConnectionString)

End Using

End Sub

Private Function GetConnectionString() As String

' To avoid storing the connection string in your code,

' you can retrieve it from a configuration file.

Return "Data Source=MSSQL1;Database=AdventureWorks;" \_

& "Integrated Security=true;"

End Function

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl58_ctl00_ctl02_code');" \o "Copy Code)

private static void OpenSqlConnection()

{

string connectionString = GetConnectionString();

using (SqlConnection connection = new SqlConnection())

{

connection.ConnectionString = connectionString;

connection.Open();

Console.WriteLine("State: {0}", connection.State);

Console.WriteLine("ConnectionString: {0}",

connection.ConnectionString);

}

}

static private string GetConnectionString()

{

// To avoid storing the connection string in your code,

// you can retrieve it from a configuration file.

return "Data Source=MSSQL1;Initial Catalog=AdventureWorks;"

+ "Integrated Security=true;";

}