.NET Framework 4

**DataAdapter Parameters (ADO.NET)**

The [DbDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.common.dbdataadapter.aspx) has four properties that are used to retrieve data from and update data to the data source: the [SelectCommand](http://msdn.microsoft.com/en-us/library/system.data.common.dbdataadapter.selectcommand.aspx) property returns data from the data source; and the [InsertCommand](http://msdn.microsoft.com/en-us/library/system.data.common.dbdataadapter.insertcommand.aspx) , [UpdateCommand](http://msdn.microsoft.com/en-us/library/system.data.common.dbdataadapter.updatecommand.aspx), and [DeleteCommand](http://msdn.microsoft.com/en-us/library/system.data.common.dbdataadapter.deletecommand.aspx) properties are used to manage changes at the data source. The **SelectCommand** property must be set before you call the **Fill** method of the **DataAdapter**. The **InsertCommand**, **UpdateCommand**, or **DeleteCommand** properties must be set before the **Update** method of the **DataAdapter** is called, depending on what changes were made to the data in the [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx). For example, if rows have been added, the **InsertCommand** must be set before you call **Update**. When **Update** is processing an inserted, updated, or deleted row, the **DataAdapter** uses the respective **Command** property to process the action. Current information about the modified row is passed to the **Command** object through the **Parameters** collection.

When you update a row at the data source, you call the UPDATE statement, which uses a unique identifier to identify the row in the table be updated. The unique identifier is typically the value of a primary key field. The UPDATE statement uses parameters that contain both the unique identifier and the columns and values to be updated, as shown in the following Transact-SQL statement.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

UPDATE Customers SET CompanyName = @CompanyName

WHERE CustomerID = @CustomerID

|  |
| --- |
| **NoteNote** |
| The syntax for parameter placeholders depends on the data source. This example shows placeholders for a SQL Server data source. Use question mark (?) placeholders for [System.Data.OleDb](http://msdn.microsoft.com/en-us/library/system.data.oledb.aspx) and [System.Data.Odbc](http://msdn.microsoft.com/en-us/library/system.data.odbc.aspx) parameters. |

In this Visual Basic example, the **CompanyName** field is updated with the value of the @CompanyName parameter for the row where **CustomerID** equals the value of the @CustomerIDparameter. The parameters retrieve information from the modified row using the [SourceColumn](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlparameter.sourcecolumn.aspx) property of the [SqlParameter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlparameter.aspx) object. The following are the parameters for the previous sample UPDATE statement. The code assumes that the variable *adapter* represents a valid [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.aspx) object.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl14_code');" \o "Copy Code)

adapter.Parameters.Add( \_

"@CompanyName", SqlDbType.NChar, 15, "CompanyName")

Dim parameter As SqlParameter = \_

adapter.UpdateCommand.Parameters.Add("@CustomerID", \_

SqlDbType.NChar, 5, "CustomerID")

parameter.SourceVersion = DataRowVersion.Original

The **Add** method of the **Parameters** collection takes the name of the parameter, the data type, the size (if applicable to the type), and the name of the [SourceColumn](http://msdn.microsoft.com/en-us/library/system.data.common.dbparameter.sourcecolumn.aspx) from the **DataTable**. Notice that the [SourceVersion](http://msdn.microsoft.com/en-us/library/system.data.common.dbparameter.sourceversion.aspx) of the @CustomerID parameter is set to **Original**. This guarantees that the existing row in the data source is updated if the value of the identifying column or columns has been changed in the modified [DataRow](http://msdn.microsoft.com/en-us/library/system.data.datarow.aspx). In that case, the **Original** row value would match the current value at the data source, and the **Current** row value would contain the updated value. The **SourceVersion** for the @CompanyName parameter is not set and uses the default, **Current** row value.

|  |
| --- |
| **NoteNote** |
| For both the **Fill** operations of the **DataAdapter** and the **Get** methods of the **DataReader**, the .NET Framework type is inferred from the type returned from the .NET Framework data provider. The inferred .NET Framework types and accessor methods for Microsoft SQL Server, OLE DB, and ODBC data types are described in [Data Type Mappings in ADO.NET](http://msdn.microsoft.com/en-us/library/4e5xt97a.aspx). |

http://i.msdn.microsoft.com/Global/Images/clear.gif Parameter.SourceColumn, Parameter.SourceVersion

The **SourceColumn** and **SourceVersion** may be passed as arguments to the **Parameter** constructor, or set as properties of an existing **Parameter**. The **SourceColumn** is the name of the [DataColumn](http://msdn.microsoft.com/en-us/library/system.data.datacolumn.aspx) from the [DataRow](http://msdn.microsoft.com/en-us/library/system.data.datarow.aspx) where the value of the **Parameter** will be retrieved. The **SourceVersion** specifies the **DataRow** version that the **DataAdapter** uses to retrieve the value.

The following table shows the [DataRowVersion](http://msdn.microsoft.com/en-us/library/system.data.datarowversion.aspx) enumeration values available for use with **SourceVersion**.

|  |  |
| --- | --- |
| **DataRowVersion Enumeration** | **Description** |
| **Current** | The parameter uses the current value of the column. This is the default. |
| **Default** | The parameter uses the **DefaultValue** of the column. |
| **Original** | The parameter uses the original value of the column. |
| **Proposed** | The parameter uses a proposed value. |

The **SqlClient** code example in the next section defines a parameter for an [UpdateCommand](http://msdn.microsoft.com/en-us/library/system.data.common.dbdataadapter.updatecommand.aspx) in which the **CustomerID** column is used as a **SourceColumn** for two parameters: *@CustomerID* (SET CustomerID = @CustomerID), and *@OldCustomerID* (WHERE CustomerID = @OldCustomerID). The *@CustomerID* parameter is used to update the **CustomerID** column to the current value in the **DataRow**. As a result, the **CustomerID** **SourceColumn** with a **SourceVersion** of **Current** is used. The *@OldCustomerID* parameter is used to identify the current row in the data source. Because the matching column value is found in the **Original** version of the row, the same **SourceColumn** (**CustomerID**) with a **SourceVersion** of **Original** is used.

http://i.msdn.microsoft.com/Global/Images/clear.gif Working with SqlClient Parameters

The following example demonstrates how to create a [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.aspx) and set the [MissingSchemaAction](http://msdn.microsoft.com/en-us/library/system.data.common.dataadapter.missingschemaaction.aspx) to AddWithKey in order to retrieve additional schema information from the database. The [SelectCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.selectcommand.aspx), [InsertCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.insertcommand.aspx), [UpdateCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.updatecommand.aspx), and [DeleteCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.deletecommand.aspx) properties set and their corresponding [SqlParameter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlparameter.aspx) objects added to the [Parameters](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.parameters.aspx) collection. The method returns a **SqlDataAdapter** object.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl74_ctl00_ctl09_code');" \o "Copy Code)

Public Function CreateSqlDataAdapter( \_

ByVal connection As SqlConnection) As SqlDataAdapter

Dim adapter As SqlDataAdapter = New SqlDataAdapter

adapter.MissingSchemaAction = MissingSchemaAction.AddWithKey

' Create the commands.

adapter.SelectCommand = New SqlCommand( \_

"SELECT CustomerID, CompanyName FROM CUSTOMERS", connection)

adapter.InsertCommand = New SqlCommand( \_

"INSERT INTO Customers (CustomerID, CompanyName) " & \_

"VALUES (@CustomerID, @CompanyName)", connection)

adapter.UpdateCommand = New SqlCommand( \_

"UPDATE Customers SET CustomerID = @CustomerID, CompanyName = " & \_

"@CompanyName WHERE CustomerID = @oldCustomerID", connection)

adapter.DeleteCommand = New SqlCommand( \_

"DELETE FROM Customers WHERE CustomerID = @CustomerID", connection)

' Create the parameters.

adapter.InsertCommand.Parameters.Add("@CustomerID", \_

SqlDbType.Char, 5, "CustomerID")

adapter.InsertCommand.Parameters.Add("@CompanyName", \_

SqlDbType.VarChar, 40, "CompanyName")

adapter.UpdateCommand.Parameters.Add("@CustomerID", \_

SqlDbType.Char, 5, "CustomerID")

adapter.UpdateCommand.Parameters.Add("@CompanyName", \_

SqlDbType.VarChar, 40, "CompanyName")

adapter.UpdateCommand.Parameters.Add("@oldCustomerID", \_

SqlDbType.Char, 5, "CustomerID").SourceVersion = \_

DataRowVersion.Original

adapter.DeleteCommand.Parameters.Add("@CustomerID", \_

SqlDbType.Char, 5, "CustomerID").SourceVersion = \_

DataRowVersion.Original

Return adapter

End Function

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl74_ctl00_ctl10_code');" \o "Copy Code)

public static SqlDataAdapter CreateSqlDataAdapter(SqlConnection connection)

{

SqlDataAdapter adapter = new SqlDataAdapter();

adapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;

// Create the commands.

adapter.SelectCommand = new SqlCommand(

"SELECT CustomerID, CompanyName FROM CUSTOMERS", connection);

adapter.InsertCommand = new SqlCommand(

"INSERT INTO Customers (CustomerID, CompanyName) " +

"VALUES (@CustomerID, @CompanyName)", connection);

adapter.UpdateCommand = new SqlCommand(

"UPDATE Customers SET CustomerID = @CustomerID, CompanyName = @CompanyName " +

"WHERE CustomerID = @oldCustomerID", connection);

adapter.DeleteCommand = new SqlCommand(

"DELETE FROM Customers WHERE CustomerID = @CustomerID", connection);

// Create the parameters.

adapter.InsertCommand.Parameters.Add("@CustomerID",

SqlDbType.Char, 5, "CustomerID");

adapter.InsertCommand.Parameters.Add("@CompanyName",

SqlDbType.VarChar, 40, "CompanyName");

adapter.UpdateCommand.Parameters.Add("@CustomerID",

SqlDbType.Char, 5, "CustomerID");

adapter.UpdateCommand.Parameters.Add("@CompanyName",

SqlDbType.VarChar, 40, "CompanyName");

adapter.UpdateCommand.Parameters.Add("@oldCustomerID",

SqlDbType.Char, 5, "CustomerID").SourceVersion =

DataRowVersion.Original;

adapter.DeleteCommand.Parameters.Add("@CustomerID",

SqlDbType.Char, 5, "CustomerID").SourceVersion =

DataRowVersion.Original;

return adapter;

}

http://i.msdn.microsoft.com/Global/Images/clear.gif OleDb Parameter Placeholders

For the [OleDbDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdataadapter.aspx) and [OdbcDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.odbc.odbcdataadapter.aspx) objects, you must use question mark (?) placeholders to identify the parameters.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl75_ctl00_ctl02_code');" \o "Copy Code)

Dim selectSQL As String = \_

"SELECT CustomerID, CompanyName FROM Customers " & \_

"WHERE CountryRegion = ? AND City = ?"

Dim insertSQL AS String = \_

"INSERT INTO Customers (CustomerID, CompanyName) VALUES (?, ?)"

Dim updateSQL AS String = \_

"UPDATE Customers SET CustomerID = ?, CompanyName = ? " & \_

WHERE CustomerID = ?"

Dim deleteSQL As String = "DELETE FROM Customers WHERE CustomerID = ?"

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl75_ctl00_ctl03_code');" \o "Copy Code)

string selectSQL =

"SELECT CustomerID, CompanyName FROM Customers " +

"WHERE CountryRegion = ? AND City = ?";

string insertSQL =

"INSERT INTO Customers (CustomerID, CompanyName) " +

"VALUES (?, ?)";

string updateSQL =

"UPDATE Customers SET CustomerID = ?, CompanyName = ? " +

"WHERE CustomerID = ? ";

string deleteSQL = "DELETE FROM Customers WHERE CustomerID = ?";

The parameterized query statements define which input and output parameters must be created. To create a parameter, use the **Parameters.Add** method or the **Parameter** constructor to specify the column name, data type, and size. For intrinsic data types, such as **Integer**, you do not have to include the size, or you can specify the default size.

The following code example creates the parameters for a SQL statement and then fills a **DataSet**.

http://i.msdn.microsoft.com/Global/Images/clear.gif OleDb Example

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl76_ctl00_ctl00_code');" \o "Copy Code)

' Assumes that connection is a valid OleDbConnection object.

Dim adapter As OleDbDataAdapter = New OleDbDataAdapter

Dim selectCMD AS OleDbCommand = New OleDbCommand(selectSQL, connection)

adapter.SelectCommand = selectCMD

' Add parameters and set values.

selectCMD.Parameters.Add( \_

"@CountryRegion", OleDbType.VarChar, 15).Value = "UK"

selectCMD.Parameters.Add( \_

"@City", OleDbType.VarChar, 15).Value = "London"

Dim customers As DataSet = New DataSet

adapter.Fill(customers, "Customers")

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl76_ctl00_ctl01_code');" \o "Copy Code)

// Assumes that connection is a valid OleDbConnection object.

OleDbDataAdapter adapter = new OleDbDataAdapter();

OleDbCommand selectCMD = new OleDbCommand(selectSQL, connection);

adapter.SelectCommand = selectCMD;

// Add parameters and set values.

selectCMD.Parameters.Add(

"@CountryRegion", OleDbType.VarChar, 15).Value = "UK";

selectCMD.Parameters.Add(

"@City", OleDbType.VarChar, 15).Value = "London";

DataSet customers = new DataSet();

adapter.Fill(customers, "Customers");

http://i.msdn.microsoft.com/Global/Images/clear.gif Odbc Parameters

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl77_ctl00_ctl00_code');" \o "Copy Code)

' Assumes that connection is a valid OdbcConnection object.

Dim adapter As OdbcDataAdapter = New OdbcDataAdapter

Dim selectCMD AS OdbcCommand = New OdbcCommand(selectSQL, connection)

adapter.SelectCommand = selectCMD

' Add Parameters and set values.

selectCMD.Parameters.Add("@CountryRegion", OdbcType.VarChar, 15).Value = "UK"

selectCMD.Parameters.Add("@City", OdbcType.VarChar, 15).Value = "London"

Dim customers As DataSet = New DataSet

adapter.Fill(customers, "Customers")

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl77_ctl00_ctl01_code');" \o "Copy Code)

// Assumes that connection is a valid OdbcConnection object.

OdbcDataAdapter adapter = new OdbcDataAdapter();

OdbcCommand selectCMD = new OdbcCommand(selectSQL, connection);

adapter.SelectCommand = selectCMD;

//Add Parameters and set values.

selectCMD.Parameters.Add("@CountryRegion", OdbcType.VarChar, 15).Value = "UK";

selectCMD.Parameters.Add("@City", OdbcType.VarChar, 15).Value = "London";

DataSet customers = new DataSet();

adapter.Fill(customers, "Customers");

|  |
| --- |
| **NoteNote** |
| If a parameter name is not supplied for a parameter, the parameter is given an incremental default name of Parameter*N,* starting with "Parameter1". We recommend that you avoid the Parameter*N* naming convention when you supply a parameter name, because the name that you supply might conflict with an existing default parameter name in the **ParameterCollection**. If the supplied name already exists, an exception is thrown. |